2016

Augmented UAS navigation in GPS denied terrain environments using synthetic vision

Teng Wang
Iowa State University

Augmented UAS navigation in GPS denied terrain environments using synthetic vision

by

Teng Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:

Arun K. Somani, Major Professor

Joseph Zambreno

Nicola Elia

Namrata Vaswani

Peter Sherman

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Arun K. Somani for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr.Joseph Zambreno, Dr.Nicola Elia, Dr.Namrata Vaswani and Dr.Peter Sherman. I would additionally like to thank my parents for their supports throughout my whole graduate career.

ABSTRACT

GPS is a critical sensor for Unmanned Aircraft Systems (UASs) navigation due to its accuracy, global coverage, and small hardware footprint. However, GPS is subject to interruption or denial due to signal blockage or RF interference. In such a case, position, velocity and altitude (PVA) performance from other inertial and air data sensor is not sufficient for UAS platforms to continue their primary missions, especially for small UASs.

Recently, image-based navigation has been developed to address GPS outages for UASs, since most of these platforms already include a camera as standard equipage. This thesis develops a novel, automated UAS navigation augmentation scheme, which utilizes publicly available open source geo-referenced vector map data, in conjunction with real-time optical imagery from on-board monocular camera to augment UAS navigation in GPS denied terrain environments. The main idea is to analyze and use terrain drainage patterns for GPS-denied navigation of small UASs, such as ScanEagle, utilizing a down-looking fixed monocular imager. We leverage the analogy between terrain drainage patterns and human fingerprints, to match local drainage patterns to GPU (Graphics Processing Unit) rendered parallax occlusion maps of geo-registered radar returns (GRRR). The matching occurs in real-time. GRRR is assumed to be loaded on-board the aircraft pre-mission, so as not to require a scanning aperture radar during the mission. Once a successful match is made, using a known lens model a final PVA solution can be obtained from the extrinsic matrix of the camera [1]. Our approach allows extension of UAS missions to GPS denied terrain areas, with no assumption of human-made geographic objects.

We study the influence of granularity of terrain drainage patterns on performance of our minutiae-based terrain matching approach. Based on experimental observations, we conclude that our approach delivers a satisfactory performance. We identify the conditions to achieve the desired performance for the input images based on UAS flight altitudes.

CHAPTER 1. GPS DENIED UAS NAVIGATION

1.1 Problem

An Unmanned Aircraft System (UAS), such as drone, is defined as an aircraft without on-board human pilot. Over the past decade, proliferation of small UASs for militiary uses has led to rapid technological advancement. This advancement provides UAS tremendous potential to create new applications in various research areas. These applications range from scientific data collection [2, 3, 5, 4], to provision of military reconnaissance and intelligence gathering as described in [6, 7, 8, 9, 10, 11].

Accurate position information is an important prerequisite for the effective use of UASs. These UAS platforms require accurate and reliable positioning data for guidance and situational awareness. Today various sensor data are utilized to compute the PVA state of UASs, including inertial sensor, barometric altimeter, 3D magnetic sensor and more [12]. GPS (Global Positioning System) is typically the primary source of reliable position information due to its accuracy, global coverage, and small hardware footprint [13, 14]. However, GPS is subject to interruption or denial due to signal blockage or RF interference, such as through canyons or under forest canopy. When GPS is not available, PVA performance from other inertial and air data sensors is no longer sufficient. These sensor equipages integrate the PVA state over time, which results in cumulative measurement error. This degraded position performance is typically not precise enough for UAS platforms to continue their primary missions, especially for small UASs. These small UAS platforms are typically not equipped with high-end navigation components which would provide higher GPS availability as well as better dead-reckoning performance in the absence of GPS.

To overcome the problem, cutting-edge systems are developed using concepts of hybrid

navigation, fusing in all available real-time navigation data, such as radar altimeters, passive imaging sensors, and digital elevation map. Recently image-based navigation algorithms have been proposed to address GPS outage for UASs [17, 18, 20, 21, 22, 23], given that most of UAS platforms already include a camera as standard equipage. Performing navigation with real-time aerial images requires georeferenced data, either images or landmarks as a reference. Georeferenced imagery is readily available, but requires a large amount of storage. A collections of discrete landmarks instead are compact, but must be generated by preprocessing. An alternative, compact source of georeferenced data having large coverage area, is open source vector maps, from which meta-objects can be extracted for matching against real-time camera acquired images. For terrain environment, we present a novel, automated UAS navigation scheme, which utilizes publicly available open source geo-referenced vector map data, such as U. S. Geological Survey (USGS), in conjunction with real-time optical imagery from an on-board monocular camera to augment UAS navigtaion in GPS-denied environments.

A Geographic Information System (GIS) is a computerized database management system, which captures, stores, manipulates, analyzes, manages, and presents all types of geographical data [15]. GIS is layer based, with each layer representing one common feature, as shown in Fig. 1.1. For example one layer is for buildings, another for roads, and so on. In our research, we mainly focus on the terrain layer of GIS. That is to say, we work on UAS navigation augmentation in GPS denied terrain environments.

Figure 1.1: Layer-based GIS, with each layer representing one common feature. While not all layers are useful, a good number of them are, including Structure, Hydrography, Transportation and Elevation layers. In our system, we only consider the single elevation layer from USGS.

1.2 Overview of Literatures on Vision Based UAS Navigation

Recently different vision-based navigation algorithms have been developed to address GPS outage for UASs. Since most of UAS platforms already include a camera as standard equipage, vision-based navigation does not require much additional hardware or payload. It is worth mentioning that although video cameras can also be "put out" by some adverse weather conditions, such as heavy cloud cover, vision-based navigation is still an attractive supplement to GPS due to the advantages described above. In practice one challenge of vision-based navigation is how to control error accumulation during flight.

1.2.1 Related Work: Landmark Based Approach

A landmark based navigation approach is proposed in [20]. In this approach, each successive image pair is utilized to reconstruct a 3D terrain map through stereoscopic analysis. The position of the aircraft is estimated by matching this reconstructed terrain map to a pre-stored, 3D digital elevation map (DEM). This approach is a two-dimensional extension of the DTS approach as described in [19].

A more recent work is presnted in [21]. It explores practical implementation as well as system integration issues. In this approach, the system is divided into two parts: relative position and absolute position estimation. Relative position estimation extracts relative displacement from two successive aerial images and computes the current position of the aircraft by accumulating relative displacement estimates. The position error from relative position estimation is compensated by absolute position estimation. The absolute position estimation is achieved by the following two approaches: (1) matching aerial image directly to reference images when the aerial image contains distinct geometric structures such as roads and large buildings; and (2) matching recovered elevation map (REM) with pre-stored DEM information in mountain regions without artificial structures. Since reconstruction and matching are performed each time that a position estimate is updated, the approaches in [20] and [21] keep correcting position error during flight. However, since stereopsis is often a difficult inverse problem, the terrain reconstruction process itself is potentially error-prone. This can cause position estimation error.

Recently, some novel image-based navigation algorithms have been developed. These approaches match real-time camera images to template landmarks, instead of DEM information, in storage to achieve precise navigation. For example, Michaelsen et. al [22, 23] developed a testbed to utilize known constructive features and patterns of salient man-made objects from handbook on infrastructure construction or thesauri. The declarative knowledge is presented as production system, and after the instances of most primitive object types are extracted from aerial image, the hypothesis driven parser is used to control the search.

Celik et. al [17] developed a system called "Meta Image Navigation Augmenters (MINA)", which utilizes real-time camera images in conjunction with open source map data to augment UAS navigation. MINA considered the transportation layer of GIS from OSM (OpenStreetMap). In this system, visual significance of objects, with precise position in 3D world coordinate system, are analyzed. Only visually distinguishing objects are rendered for navigation augmentation purpose. The features of visually distinguishing objects are pre-stored in the training sets as landmarks. During a flight in GPS-denied environments, matching approaches that work with geometric shapes and parametric curves including weighted K-nearest neighbor classifier, thin plate spline transform and principal component analysis, are applied to match aerial images against templates in the training sets. The matching result is used to augment UAS navigation. This system is proved to perform well in urban areas. It keeps correcting position error during flight. However, this approach could not be directly extended to terrain areas, as the representation and matching methods for artificial structures cannot be applied to terrain.

1.2.2 Different Approaches

Merhav and Bresler [24, 25] developed two position estimators for UAS navigation. The first estimator obtains the position information of UASs by integrating instantaneous velocity estimates. The instantaneous velocity is estimated using inter-frame displacement of the input image sequence and the airplane's height readings. The second estimator obtains the position information through an extended Kalman filter (EKF). For the EKF, the airplane's position and velocity are defined as its states, and the inter-frame displacement measurements are

considered as observation data. Using a theoretical analysis, Merhav and Bresler show that the first estimator suffers from error accumulation due to integration operations while EKF that locks on to the available terrain information does not. However, Merhav's and Bresler's paper does not report performance of both estimators on video data. In addition, the motion model used in EKF is rather simple: only the translation motion (no rotation) is considered.

Lerner et al. [26] propose an approach that uses the DEM directly to generate a constraint between the camera's pose and ego-motion. This leads to a set of nonlinear equations from which in principle one can obtain UAS's pose as well as 3D motion. The focus of the paper was on expressions of estimation error measures as well as their Monte Carlo simulations, since these measures are functions of various parameters and are too complex to evaluate analytically. However, this paper did not describes how to solve these nonlinear equations in detail.

In other works [27, 28, 29], researchers adapt the SLAM (simultaneous localization and mapping) technique to vision-based robot navigation. SLAM was developed for a robot to estimate its own position as well as the positions of a set of landmarks. This is achieved by using an EKF, which utilizes range measurements from the robot to the landmarks as the observation data. One problem with SLAM is that vision sensors generally do not provide range information, additional data is therefore required to generate the needed range measurements. For example, the technique in [27, 29] requires man-made ground landmarks with knowns sizes, and the technique in [28] requires some ground landmarks with known 3D coordinates. However, these techniques are not scalable to large area, because they use EKF and is n^2 computationally expensive.

He at. al [30] proposed a hierarchical framework to deal with uncertainty and noises in motion field analysis. In this approach, images are decomposed into structural blocks containing distinctive features and non-structural blocks. Motion estimation is done for each structural block through feature tracking. A reliable value is assigned to each estimated motion vector, and only motion vectors with higher reliability are used for camera motion estimation. One problem with this approach is that it couldn't be applied to terrain area which might not contain structural objects.

Saripalli [31] developed an algorithm that uses cheap accelerometers and gyroscopes, and

combines them with visual updates in an EKF framework to provide precise orientation information. The states of EKF consist of the position, velocity, altitude of UAS, gyroscope biases, and magnitude of the gravity vector. Vehicle height and motion, which are estimated by tracking corner features between consecutive frames via stereo process, are considered as observation data of KEF. This approach only uses a loosely-coupled kalman filter framework to include the vision updates, instead of incorporating the feature tracks themselves in the kalman filter framework, causing an estimation error.

Recently, Zhang et. al [32] proposed an approach that uses the terrain DEM directly to estimate the position and orientation of UASs. In this approach the position and orientation estimation is formulated as a tracking problem and solved by using an extended Kalman filter (EKF). For the EFK, the UAS's position, orientation, velocity and angular velocity are defined as its states, and the small number of visually salient pixels (feature points) from the aerial images are defined as the observation data. The state and observation models of the EKF are established based on an analysis of the imaging geometry of the UAS's video camera in connection with a DEM of the flight area. One problem with this approach is that one needs to consider complex aircraft model. Typically the dynamics of the helicopter is described using a conventional six-degree-of-freedom rigid body model, and therefore EKF requires high computational overhead for derivations.

1.3 Motivation

Based on the discussion in Section 1.2, we conclude that specific type of vision-based navigation algorithm, which match the aerial image (recovered elevation map) against landmarks (digital elevation map) pre-stored in databases to estimate position of UAS, are attractive as they keep correcting PVA state error during flight.

For terrain matching, one alternative is to match the recovered terrain elevation map against pre-stored digital elevation model information. Another alternative is to store map data in database and generate life-like appearances of terrain dynamically during flight. Aerial images can be matched directly against map data-rendered terrain images via feature extraction and matching processes to estimate position of UAS. The terrain reconstruction process itself is

potentially error-prone. This is because stereopsis is often a difficult inverse problem, and this can cause position estimation error. We therefore propose a novel approach that utilizes aerial images combined with open source map data to augment UAS navigation in GPS-challenged terrain environments.

1.4 Research Contributions

This work build a fully automated system of machine vision algorithms for map-aided navigation of UASs, such as ScanEagle, in GPS challenged terrain environments. The system can be divided into five major parts: (1) Imaging Component, (2) MetaMap Component, (3) Landmark Render Component, (4) Feature Extraction Component, and (5) Matching component, as shown in Fig. 1.2.

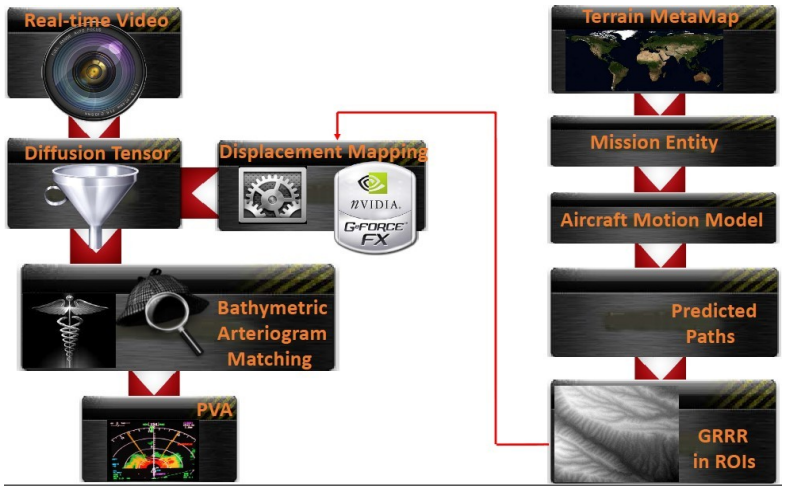


Figure 1.2: Functional diagram of our map-aided UAS navigation system

Imaging components acquire real-time image data. Metamap component loads open source map data on-board the aircraft pre-mission, and output map data of landmarks in ROIs (Regions of Interests) during GPS denied flight. Landmark Render component collaborates with a metamap to generate landmarks, i.e., life-like appearance of terrain areas with distinguishing features that the aircraft is expected to encounter. These acquired and rendered images are enhanced and filtered to emphasize certain terrain features by Feature Extraction component. Matching component then matches aerial images to landmarks to find a match. Once a suc-

cessful match is made, with a known lens model a final PVA solution can be obtained from the extrinsic matrix of the camera. To achieve this goal, we make the following contributions:

1. **Landmark Render Component**
   - Choose an appropriate open source map data for terrain recreation.
   - Develop a real-time terrain generation technique based on per-pixel displacement mapping.
2. **Feature Extraction Component**
   - Identify, develop and analyze a novel minutiae feature which are minor details on drainage patterns, for terrain recognition purpose.
   - Design, develop and analyze a series of filters to extract drainage patterns from terrain images and identify minutiae on them.
3. **Matching Component**
   - Identify two shape descriptors to describe neighborhood similarity between different minutiae.
   - Develop a matching process to identify minutiae pairings using shape descriptors from two input minutiae sets.
   - Define and identify a criterion to measure similarity of two input terrain images based on number of minutiae pairings.
4. **Granularity of Drainage Patterns**
   - Study the influences of granularity of drainage patterns on performance of minutiae-based terrain recognition approach.
   - Develop a requirement for minimum minutiae number in two input terrain images in order to ensure a satisfactory recognition performance.

1.5 Thesis Organization

This chapter starts by describing the challenges of UAS navigation in GPS challenged environments. We then discuss current approaches for UAS navigation augmentation in GPS denied environments. The rest of this document is organized as follows.

Chapter 2 discuss the selection of an appropriate map source for terrain rendering, and presents a real-time terrain generation technique using per-pixel displacement mapping in detail. In Chapter 3, we define a novel minutiae feature for terrain recognition purpose and design a series of filters to extract drainage patterns as well as identify minutiae from raw terrain images. Following this, Chapter 4 descibes the minutiae-based terrain matching process to measure similarity between two input terrain blocks. Subsequently, we study the influence of granularity of drainage patterns, i.e., flight heights, on performance of our minutiae-based terrain recognition approach in Chapter 5. Finally, Chapter 6 covers the concluding remarks and future work.

CHAPTER 2. REAL-TIME TERRAIN GENERATION

In this chapter, we discuss open source map selection, and describe in detail the generation of life-like appearance of terrain from open source map data using advanced 3D computer graphics techniques.

2.1 MetaMap

2.1.1 Map Source Selection

GIS captures, stores, manipulates, analyzes, manages, and presents all types of geographical data. GIS is layer based, with each layer representing one common feature, as shown in Fig. 1.1. Terrain and bathymetry are such layers. Our UAS navigation system interprets GIS data at terrain layer to render metamaps, which are later used for terrain recognition purpose.

Natural layers of GIS are collected by professionals and come in structured forms, some of which are highly specialized to specific tasks. We considered and experimented with a variety of maps, before deciding to integrate GRRR from an open source map provider USGS into our UAS navigation system. Accelerated by the proliferation of small, affordable, and lightweight electronically scanning radar systems as well as UAS, GRRR imaging is becoming an incredible source for logistics. GRRR can be characterized as structured, heterogeneous, and scientifically collected map data with consistent amounts of completeness and standardization of resolution. GRRR has the following advantages.

- GRRR is open-cource. USGS provide it in GEOTIFF numerical format, where other providers will rather provide a raster image of the same tile. More details about GEOTIFF format can be found in [33].

- GRRR data is based on ASCII, which is both a very convenient debugging feature, and resistant to data corruption.
- While GRRR can be used in quadrilateral tiles that is not an obligation. It can be downloaded at any size, shape, or proportions, and does not have to be complete.
- GRRR does not carry any watermarks, legends, brand logos, and other artifacts to be forcibly rendered on map.
- GRRR is customizable, and this is the primary reason it can be made to suit the application. With other map providers, the map always look like how the provider wants it to look. Rendering, aliasing, coloring, compression, line-styles, and many more parameters are their proprietary style of map and cannot be modified.
- GRRR allows finer control over emphasizing particular features of interest of terrain area. For example, it can be height thresholded. Water bodies can be individually colored or removed together. Vegetation can be dynamically modeled, There are no limits.

GRRR has the following disadvantges:

- Because GRRR requires advanced technologies to generate, it is most densely available for continental U.S. In other countries, it is generally available, but the coverage and resolution varies.
- Also for above reason GRRR is only modified when major natural disasters happen, and the changes are to be considered negligible.

2.1.2 GRRR Organization

GRRR data provided by USGS are collected by Shuttle Radar Topography Mission (SRTM). SRTM is an international research project, which recorded data of the entire land mass of the earth from 60°N to 56°S using interferometric synthetic aperture radar (InSAR). It generates the most complete high-resolution digital elevation model (DEM) of the earth. More details about SRTM, including an overview of the mission as well as the DEM production, and an evaluation of the DEM accuracy, can be found in [34, 35, 36].

SRTM data are arranged into individual rasterized tiles. Each tile covers one degree of latitude by one degree of longitude. Sampling space for individual points in both latitude and longitude can be either 1/3 arcsecond, 1 arcsecond, 3 arcsecond, or 30 arcseconds, which are referred to as SRTM1/3, SRTM1, SRTM3 and SRTM30, respectively. In our application, we choose SRTM1/3 data for terrain generation within United States due to its high resolution. The higher resolution of GRRR data, the more terrain details can be captured by the map data. It is worth mentioning that SRTM1/3 is also called "10 meter" data due to the fact that one arcsecond at the equator corresponds to roughly 30 meters in horizontal extent.

SRTM1/3 data are sampled at one-third arcsecond in both latitude and longitude. The resolution of the map data is one-third arcsecond (i.e., 10m) and the data is only released over United States. Each file of one-third arcsecond tile contains 10812 rows, and each row consists of 10812 cells. That is to say, the dimension of the one-third arcsecond tile is equal to 10812×10812. Each cell stores an elevation value generated by averaging all radar returns which fall within that cell. The "average" operation is introduced to reduce the primary error from synthetic aperature radar data, which is speckle and has the characteristic of random noise. All elevations are measured in meters referenced to the WGS84/EGM96 geoid.

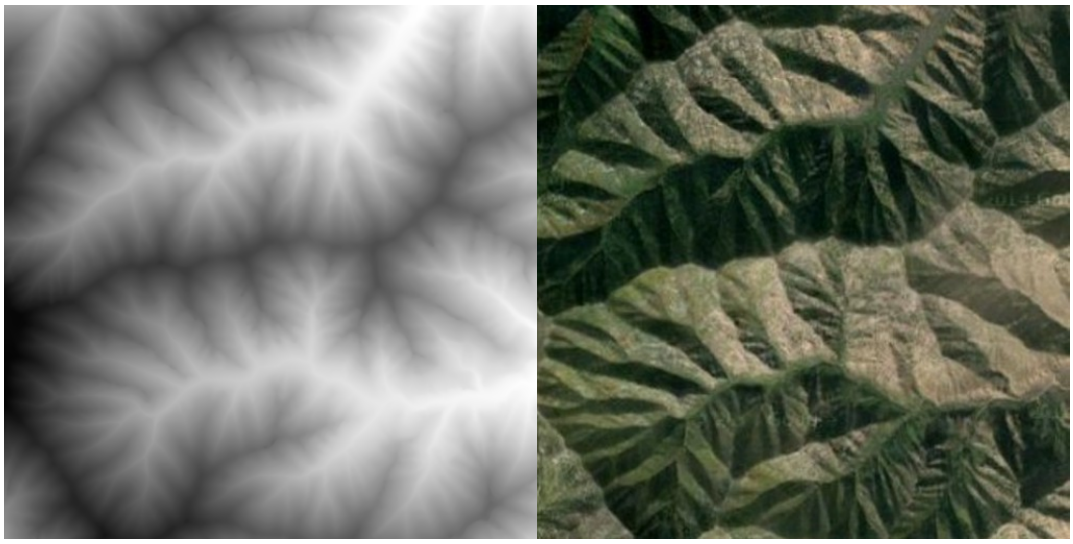
An example of SRTM data at 10m resolution is given in Fig. 2.1, which shows part of GRRR data from tile N39W111 in ESRI ASCII raster format. It starts with header information, which defines properties of the raster. Raster properities include its dimension, cell size, as well as coordinate of the origin of the raster. Here, **nrows** and **ncols** are numbers of cell rows and columns, respectively; **xllcorner** and **yllcorner** represent longitude and latitude value of the south western (i.e., lower left) corner of the tile, respectively; **cellsize** represents size of each cell in degree. The header information is followed by an array of cell elevation values (e.g., 2740.487, 2740.767 and so on) specified in space-delimited row-major order, with each row separated by a carraige return.

As shown in Fig. 2.1, tile N40W111 covers one degree of latitude by one degree of longitude, stretching from N39W111 to N40W110. It's difficult to display the whole terrain scene in detail due to its large area. We take Kings Peak within the tile as an example for explanation. To facilitate observation, we generate 24-bit grayscale DEM of Kings Peak using GRRR data from

```
ncols         10812
nrows         10812
xllcorner     -111.0005555556
yllcorner     38.99944444444
cellsize      9.2592592593e-005
NODATA_value  -9999
2740.487 2740.767 2741.734 2742.862 2743.366
2746.494 2745.314 2743.964 2742.427 2740.782
```

Figure 2.1: GRRR of tile N40W111 in ESRI ASCII raster format.

SRTM. The DEM is shown in Fig. 2.2a. Size of DEM is equal to 512 by 512 pixels. Each pixel corresponds to one terrain cell of size 10m by 10m. Here, the value of a pixel represents an elevation, instead of a luminance intensity. In detail, the brighter the gray level of a pixel, the larger the elevation value of the terrain point corresponding to this pixel. Fig. 2.2b shows the aerial image of the same terrain block from monocular camera. By comparing the DEM with the aerial image, we make the observation that networks of ridges and valleys in the terrain appear clearly on the DEM. That is to say, GRRR data from USGS-SRTM has the capability to accurately capture the ridge-valley features in this terrain area.



(a) Grayscale DEM (b) Aerial image

Figure 2.2: DEM of Kings Peak area in Utah. The DEM data in Fig. 2.2a can be found at http://nationalmap.gov/elevation.html

2.1.3 DEM Interpolation

As described in Section 2.1.2, we decide to intergate USGS-SRTM data at 10m resolution into our navigation system due to its high resolution. This data works very well for UAS applications operating at altitudes ranging from 8000 to 10000 meters. However, in some applications UASs need to operate at a relatively lower altitude, and therefore DEM data at a finer resolution are required. ArcGIS Spatial Analyst provides many options for interpolating spatial data to higher resolutions, including Inverse Distance Weighted (IDW), Ordinary Kriging and Spline. In this section, we give a brief description of these approaches and compare their performances.

Inverse Distance Weighted is one deterministic interpolation approach using a set of values at scattered sampled locations. It is based on the assumption that the elevation value at a terrain point is more influenced by values at nearby sampled locations compared to those at distant locations. IDW computes the interpolated elevation value at an unknown point using a linear-weighted sum of the values at sampled locations within search neighborhood. Mathematically, IDW assign the value $\hat{z}(x_0)$ to an unknown point x_0 using the following equation:

$$\hat{z}(x_0) = \frac{\sum_{i=1}^{n} w(d_i) \cdot z(x_i)}{\sum_{i=1}^{n} w(d_i)} \tag{2.1}$$

where x_i represents the ith sampled location; $z(x_i)$ is the measured elevation value at terrain point x_i; d_i is the distance between x_i and x_0; $w(d_i)$ is the weight assigned to point x_i, where $w(\cdot)$ is a decreasing function. The general form of weight function is $w(d) = d^{-\mu}$, where exponent μ is commonly set as 1 and 2. The use of higher exponent will create a more localized interpolator by decreaseing weights of sampling points which are far away from x_0. More details about IDW can be found in [37, 38, 46].

Kriging is another method of interpolation that use regionalized variable theory to determine the weights of known values at sampled locations. More details about mathematical description of Kriging can be found in [39, 40, 41]. There are several different types of Kriging, including Ordinary Kriging, Simple Kriging, Universal Kriging and so on. Among these different types of Kriging, **Ordinary Kriging** is the most commonly used method due to its weak stationary

requirements. It is based on the assumption that there exist constant mean over the search neighborhood around the unknown point x_0. The unknown elevtaion value $Z(x_0)$ at point x_0, as well as value $Z(x_i)$ at sampled location $x_i, i = 1, 2, \cdots, n$, are all interpreted as random variables. Ordinary Kriging aims to develop a linear estimator $\hat{Z}(x_0)$, taking in the form of

$$\hat{Z}(x_0) = \sum_{i=1}^{n} \lambda_i Z(x_i), \tag{2.2}$$

where λ_i is the weight assigned to the elevation value at sampled location x_i. The weight λ_i is chosen to ensure that the following estimation error

$$\epsilon(x_0) = Z(x_0) - \hat{Z}(x_0) \tag{2.3}$$

has zero expected value and minimal variance. In other words, Kriging gives the best linear unbiased predication of the elevation value $Z(x_0)$ at point x_0. By solving the optimization problem, we obtain that

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{bmatrix} = \begin{bmatrix} 0 & \gamma(x_1 - x_2) & \cdots & \gamma(x_1 - x_n) & 1 \\ \vdots & & & & \vdots \\ \gamma(x_n - x_1) & \gamma(x_n - x_2) & \cdots & 0 & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma(x_1 - x_0) \\ \vdots \\ \gamma(x_n - x_0) \\ 1 \end{bmatrix}, \tag{2.4}$$

where $\gamma(\cdot)$ is the variogram function, and μ represents the lagrange multiplier. Aftering obtaining weight λ_i, the interpolated elevation value at x_0 is given as follows:

$$\hat{z}(x_0) = \sum_{i=1}^{N} \lambda_i z(x_i), \tag{2.5}$$

where $z(x_i)$ represents measured elevation (i.e., observations) at sampled location x_i. More details about Ordinary Kriging can be found in [42, 43, 46].

Spline is an interpolation method that aims to fit a smooth curve to a set of measured elevations $\{z(x_i)\}, i = 1, \cdots, n$, using a spline function. The spline makes a compromise between two quite different objectives of curve fitting: (1) Accurate fitting to data at sampled locations (2) Interpolation function to be as smooth as possible. Spline acheives these two goals by developing a curve estimate which minimizes the following criterion

$$\sum_{i=0}^{n} \{z(x_i) - S(x_i)\}^2 + \alpha * I(S), \tag{2.6}$$

where $S(\cdot)$ is the spline function; $\sum_{i=0}^{n}\{z(x_i) - S(x_i)\}^2$ represents the sum of deviation from measured elevations at sampled locations; $I(S)$ measures roughness of the interpolation curve. α is the parameter controlling the trade-off between fitting accuracy and smoothness of the curve. More details about spline interpolation can be found in [44, 45]. With fitting curve available, the estimated elevation at non-sampling point x_0 is calculated as

$$\hat{z}(x_0) = S(x_0) \tag{2.7}$$

Guarneri et al [46] compared accuracy of these three interpolation methods using DEM data from the Quachita Mountains in central Arkansas. In detail, they employ each method to interpolate the 10m DEM to 5m, 2.5m, and 1m resolutions. For each resolution, they compare the absolute mean differences of three methods using surveyed control point. Experiment results prove that there is slight difference in the accuracy between these interpolation methods. Given the similarity in accuracy, processing time becomes a more important factor in deciding which method to employ in our application. Based on the observation in [46] that IDW is the fastest among these three interpolators, we prefer IDW interpolation to generate finer resolution DEM of terrain when needed.

2.2 GPU Render

Render component uses the GRRR data to build a digital elevation model (DEM) of the terrain, and renders them to realistic model.

2.2.1 Pixel Displacement Mapping

Computers represent 3D objects with polygons. The more polygons, the more details can be represented. This works very well for geometric volumes such as cube where few polygons can be used to represent very large objects. However, when it comes to a whole mountain it takes billions of polygons to represent it in 3D as in much detail as it is seen from the on-board aircraft camera. This is a huge computational burden.

Using graphic processors, however we can use fewer polygons (hundreds) to represent a 3D mountain, where simulation of lighting of bumps and dents is interpolated according to

natural models. In other words, we take a low resolution terrain model and statistically fill in the detail to convincingly make it look like a much higher resolution image taken from an on-board aircraft camera. We achieve this via displacement mapping. Main idea of displacement mapping is to employ a macrostructure to approximate a 3D terrain, and add gemotric details to the macrostructure surface using a height map. The height map describes the difference between the macrostructure and the true terrain model in the direction of the macrostructure normal vector. In detail, displacement mapping takes sample points and displace them along the local normal of the macrostructure surface, as illustrated in Fig. 2.3. The displaced distance is determined by the value stored in the height map at the given point. More discussion on displacement mapping can be found in [47, 48, 49, 51].

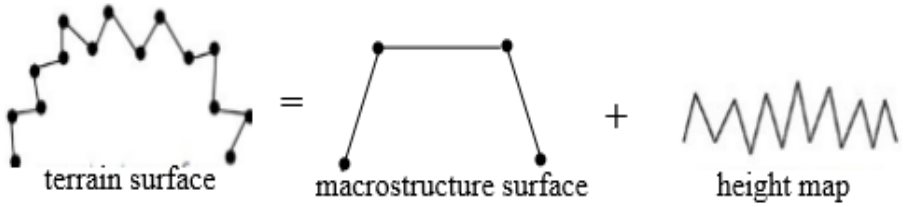


Figure 2.3: Basic idea of displacement mapping. This figure can be found in [51].

Displacement mapping can be implemented in both vertex and pixel shader. In case of per-vertex displacment mapping, the sample points for displacement refer to vertices of the original or tessellated mesh. The usual implementation of per-vertex displacement mapping iteratively tessellates a base surface (i.e., macrostructure surface), and pushes vertices out along the normal of the highly tessellated surface as described in [50]. However, there are some problems with vertex displacement methods as discussed in [51], which are described as follows.

- The number of vertices in the tessellated surface can be very high. This contradicts to the objective of displacement mapping in hardware accelerated environment, which is to reduce the number of vertex while keeping surface details.

- Vertex processing power of a graphics processor is usually smaller than pixel processing. And pixel pipelines can perform more operations per clock cycle than vertex pipeline.

- Pixel shaders are better equipped to access textures. Most graphics cards will prevent texture access within a vertex shader because it has a high performance cost.

- As vextex shader always executes once for each vertex in the terrain model, the work in vertex shader in evenly distrubuted over the entire terrain geometry, even those invisible parts, where it's not needed. However, the work in pixel shader is concentrated on nearby visible components, where it is needed most.

A minor disadvantages of displacement mapping using pixel shader is as follows.

- It is not possible to alter a pixel's screen coordinate within the shader. That means per-pixel displacement mapping cannot be used for arbitrary large displacement. This is not a major issue because displacements are almost always bounded in a terrain model.

Based on the above discussion, we prefer to implement per-pixel displacement mapping in our Render Component. In per-pixel dispalcement mapping, the samples points for displacement refer to the points corresponding to the texel centers.

2.2.2 Ray Surface Intercepting

In implementation of per-pixel displacement mapping, the macrostructure geomtery of a terrain goes through the graphic pipeline. When it arrived at fragement shader, the height map is utilized to add details to the macrostructure surface in order to make it look like the true terrain surface. However, we are not allowed to change geometry of the terrain at fragment shader, and therefore the visibility problem needs to be addressed through a ray-tracing like algorithm. It can be imagined as tracing the view ray into the height field in order to find the point which is really seen by the camera. The texture coordinate of the visible point will be used to fetch color and surface normal vector, which are later used for computing pixel intensity. Fig. 2.4 shows basic idea of per-pixel displacement mapping.

A brief mathematical description of Per-Pixel displacement mapping based on ray tracing is given as follows. More details can be found in [51]. Suppose the current fragement shader get one point of the macrostructure surface, with texture coordinate of the point equal to $[u, v]$.

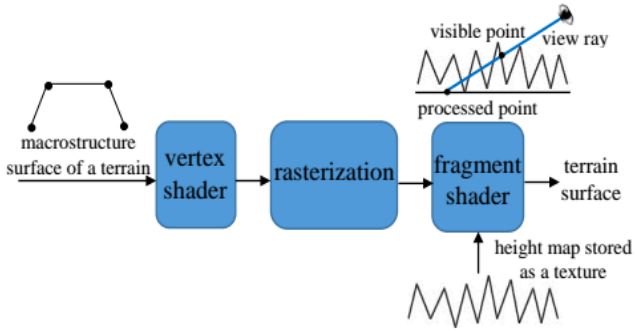


Figure 2.4: Basic idea of displacement mapping on fragment shader.

The coordinate of the processed point in tangent space can be expressed as $(u, v, 0)$. For this pixel point, we define a view ray connecting the eye and the processed point $(u, v, 0)$. The direction of this view ray is defined by view vector $\vec{V}$ in tangent space. The fragment shader program aims to find the point on the height field, which is really seen by the view ray $\vec{V}$. Considering the fact that the visible point is on the height field, its coordinate in tangent space can be expressed in the form of $(u', v', h(u', v'))$ for some unknown (u', v'). And therefore this visible point can be identified by solving the equation

$$(u', v', h(u', v')) = (u, v, 0) + \vec{V}t. \tag{2.8}$$

for some parameter t. The equation might have several solutions, i.e., several points $(u', v', h(u', v'))$ on the height field which are projected onto the same pixel. But only the intersection closest to the eye, which is called "first intersection" is of interest. Fig. 2.5 shows an example where there exist three intersections of the view ray with the height field.

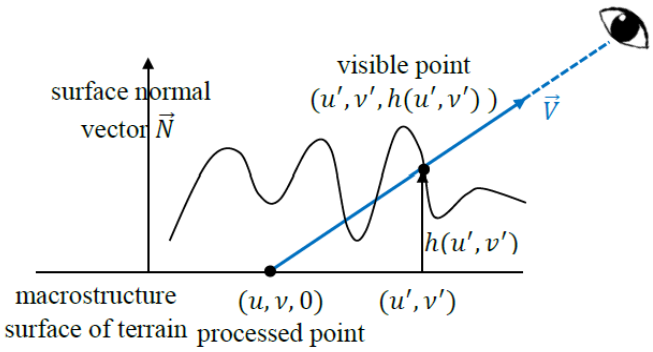


Figure 2.5: Ray tracing of the height field.

In our work, we employ the Sphere Tracing technique as described in [52] to find the visible point. This approach is attractive as it solved the aliasing problem caused by the commonly used approach of "Uniform Sampling". Sphere Tracing uses a distance map and an iterative algorithm to guarantee that the first hit will always be found. To construct a distance map, we define a distance function $\mathbf{dist}(p, S) = \mathbf{min}\{d(p, q), q \in S\}$ for any point p in texture space, where S represents a displaced terrain surface. $\mathbf{dist}(p, S)$ stores the shortest distance from p to the closest point on the surface S. The distance map for a surface S is then defined as a 3D texture that stores the value of $\mathbf{dist}(p, S)$ for each point p. More details about the algorithm of distance map computing can be found in [53].

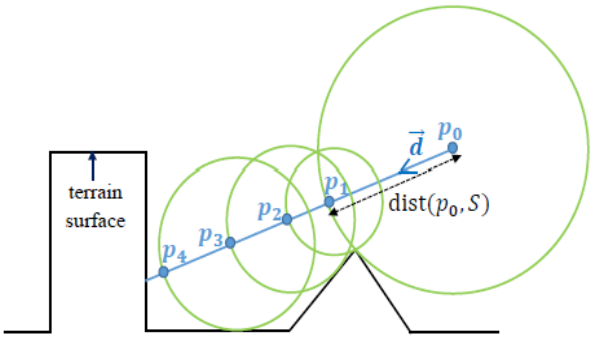


Figure 2.6: Sphere tracing of view ray with displaced surface.

This distance map gives a measure of the distance between points in texture space and the displaced surface. This distance map provides all the information to intersect a viewing ray with the displaced surface S. Suppose we have a ray with origin p_0 and normalized viewing direction vector $\vec{d}$. Using distance information $\mathrm{dist}(p_0, S)$, we define a new point $p_1 = p_0 + \mathrm{dist}(p_0, S) \times \vec{d}$. One important property of the point p_1 is that it will be outside the surface if its previous point p_0 is outside the surface. We repeat the same operation to define another new point $p_2 = p_1 + \mathrm{dist}(p_1, S) \times \vec{d}$, and so on. We make the observation that each point p_{i+1} is a little bit closer to the surface compared to the previous point p_i, as shown in Fig. 2.6. That means, we could always find the first intersection of the ray with the surface (i.e., visible point) by taking enough samples. More discussion about Sphere Tracing can be found in [52]. Once we obtain the texture coordinates of the intersection point, we compute normal-mapped lighting in tangent space.

Based on the above discussion, the whole terrain generation procedure based on per-pixel displacement mapping can be summarized as follows:

- Approximate the shape of terrain surface using triangles or quads.
- Generate heightmap of the terrain in raster format using GRRR data from USGS-SRTM.
- Compute tangent-space surface normals using the heightmap and store them in a texture.
- Compute 3D distance map using the heightmap and store it directly in a texture.
- Create a vegetation texture based on terrain elevations, and store it directly in a texture.
- Vertex Shader Program:
    1. Project vertex position into screen space and pass through the texture coordinate.
    2. Transform the eye vector into tangent space. The tangent-space eye vector is later used in the fragment shader as the direction of the ray to be traced.
    3. Transform the light vector into tangent space. The light vector is later used in the fragment shader for tangent-space normal mapping.
- Pixel Shader Program:
    1. March the view ray iteratively by querying the distance map to obtain a consecutive estimate of the distance and moving along the ray until reaching the closest intersection.
    2. Compute normal-mapped lighting by multiplying unit light vector with remapped normal vector at intersection point.

More details about Per-Pixel displacement mapping on GPU as well as its shader implementation can be found in [52].

2.3 Experiment

We utilize an NVIDIA Quadro 5100 graphics processor to achieve the rendering process. We pick Quadro instead of Geforce GTX 480M for terrain generation due to the fact that Quadro is specifically meant for scientific computing and it offers ECC RAM with FP64 IEEE 754. It is optimized for stability and performance in professional applications, like medicine. The CUDA cores can be accessed readily and it uses a Fermi architecture designed to offer higher levels of performance in CPU calculations.

We experiment the terrain generation approach based on displacement mapping over **Hualapai Peak** $(35°03'59''N, 113°45'58''W)$ area in Arizona. Image data of Hualapai Peak from on-board monocular camera is presented in Fig. 2.7. Size of the aerial image is 512 by 512 pixels, where each pixel corresponds to one terrain cell of size 10m $\times$ 10m.

Figure 2.7: Aerial Image of Hualapai Peak in Arizona.

We start the rendering process with building the DEM of the terrain, directly utilizing USGS-SRTM data at 10m resolution. 24-bit grayscale DEM of **Hualapai Peak** is presented in Fig. 2.8, where pixel intensity value is proportional to terrain elevation. The scenery uses a virtual camera fixed at zenith, and changes the terrain attitude to reflect aircraft orientation. We employ directional light model to simulate sunlight. Sun position is put in direction of $(0.5, 1.0, 1.0)$ based on the time information from UAS inertial sensor. We then compute 3D distance map based on height map and perform per-pixel displacement mapping via sphere

tracing to render the terrain. Output of GPU render is displayed at Fig. 2.9. By comparing Fig. 2.9 with Fig. 2.7, we make the observation that the rendering process has the capability to accurately capture both ridges and valleys features of Hualapai Peak.

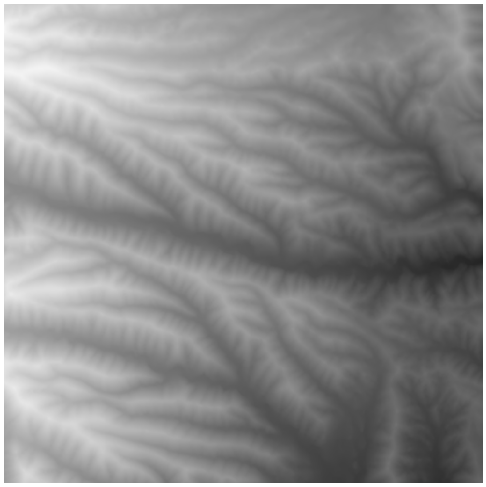


Figure 2.8: GeoTIFF data of Hualapai Peak before preprocessing. The DEM data can be found at http://nationalmap.gov/elevation.html.

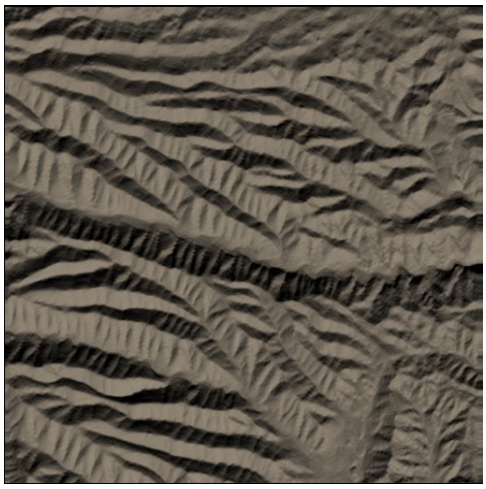


Figure 2.9: Output of GPU Render.

We further test the terrain rendering approach using displacement mapping over different terrain areas. In all experiments, image resolution is fixed, with each pixel corresponding to one terrain cell of 10m by 10m. Experiment results show that the rendering process can shade every image of size 512×512 pixels at approximately 80 frames per second, i.e., 0.0125 second/frame.

2.4 Conclusions

In this chapter, we first discuss the selection of open source map data and decide to choose GRRR data from USGS-SRTM as our map data for terrain generation purpose due to its high resolution and customizability. We then design a terrain rendering approach using displacement mapping. Experiment results show that the terrain generation approach has the capability to accurately capture drainage pattern features of terrains. In addition, the approach can render terrain image of size 512×512 pixels at average 80 frames per second.

CHAPTER 3. NATURE'S SIGNATURE

A solution to augment UAS navigation in GPS-challenged terrain environments is decidedly a minutiae-based approach due to the way nature behaves. The following principal observations of the features of nature have led to our proposed approach.

1. Acts-of-nature type terrain features, which present themselves as "detectable" to a machine vision system using some generalizable method, are that of those shaped by flow of water. Water leaves a signature that has a random distribution and can be statistically analyzed. This cannot always be precisely predicted. Common denominator in such problems, where heuristic methods are involved in matching, is that exhaustive approaches (e.g., correlation), and are impractical.

2. The positive artifacts of filtering video signals that contain water-shaped terrain using convolution kernels yield results that visually resemble minutiae formations. Spectral filters for example do not have that type of characteristics. For this reason, they would not work well with nature as they do with human-made scenes.

Observation (1) rules out correlation based approaches because the search would have to be a brute force method. Correlation itself is a second polynomial order computation and is expensive. It also would yield too many false negatives. Only under a perfect and most ideal conditions one would find a match. Observation (2) means that a method cannot guarantee optimal solution. This also means that approaches that work with geometric shapes and parametric curves have to be eliminated. While some parametric curves are found in nature, they are at a micro level and not a macro level. Mountain size objects tend not to exhibit themselves. This eliminates non-minutiae approaches.

Once we choose a minutiae-based approach, we need two sources of input that can be represented in minutiae format: (A) a processed image (from monocular camera) to be matched; and (B) templates (from GPU render) to match. The solution needs to be a relaxation of the process of finding a satisfactory correspondence matching between the two via cognitive shortcuts. We can use optimized shortcuts and come up with the most likely matching result in a probabilistic manner.

3.1 Definition Of Minutiae

Minutiae are the minor or incidental details of the terrain that are ambiguous on their own. They have spatial characteristics that can be used to uniquely identify a particular geo-location. These details are found in drainage patterns (a.k.a. creases, ridges, valleys) formed by movement of water flows, and captured in the bathymetry layer of GIS system. Note that we are not assuming presence of water; only that it had once been at that location. Some drainage patterns date as far back as the ice age, and the water that shaped that area is long gone. On the other hand, a small creek or river which still has water flow is still perfectly applicable. A few minutiae type examples are depicted in Fig. 3.1 and described below.

a. **Crease Ending:** A point at which one terrain crease terminates

b. **Bifurcation:** A point at which one terrain crease divides into two creases

c. **Dot:** An isolated crease unit whose length approximates its width in size

d. **Island:** A terrain crease slightly longer than dots, occupying space between two temporarily divergent creases

e. **Enclosure:** A single terrain crease that bifurcates and reunites

f. **Spur:** A bifurcation point at which one short crease branches off a longer crease

g. **Bridge:** A connecting crease between two parallel running terrain creases

h. **Trifurcation:** A point at which one terrain crease divides into three creases

i. **Crossing:** A point at which two terrain creases cross each other

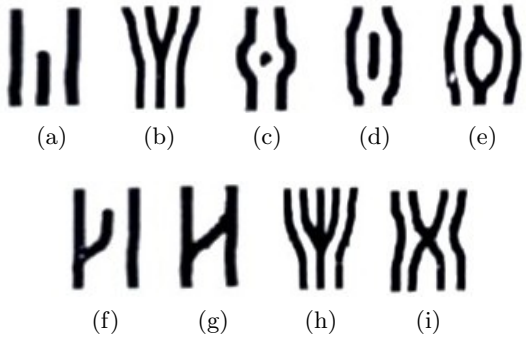


Figure 3.1: Minutiae Examples, including Crease Ending, Bifurcation, Dot, Island, Enclosure, Spur, Bridge, Trifurcation, and Crossing.

Pliability of the underlying rock to water, deposition pressure of water, slippage, roughness of the substrate are just some of the various factors that determine how water forms drainage patterns. Drainage patterns govern how minutia are formed such that no two terrains are exactly alike. Our approach is designed to bring out these signature features, while filtering out minor changes in impressions of the same terrain area recorded at different times with different ambient lightning, orientation, or seasonality.

Although minutiae are readily present in most terrain, they are not in an immediately usable form due to the ambient lighting, color, shadows, and many other factors that they are usually hidden within. As minutiae appear as minor details in drainage patterns, we first employ diffusion filtering followed by MLSEC operator, to expose drainage patterns.

3.2 Drainage Pattern Extraction

3.2.1 Diffusion Filtering

Because minutiae appear as minor details in drainage patterns, we first propose a "crease-ness" measure to extract drainage patterns. A pre-filtering step is required to reguralize the raw terrain image in order to obtain stable and meaningful crease result. We employ an anisotropic nonlinear diffusion process as described in [55] to achieve this. The advantages of this filtering process are that while crease features are enhanced, artifacts are not created and junctions are

not interrupted. A brief discussion of the diffusion filtering based on structure tensor is given as below. The main idea of the diffusion filtering is to construct a coarse creaseness measure, and control the diffusion strength in the presence of ridges and valleys (i.e., enhancing diffusion along the crease while avoiding smoothing across crease). For that, we consider *normalized Hessian matrix* H of a two-dimensional image I such that

$$H = \frac{1}{\sqrt{1+\|\nabla I\|^2}} \begin{pmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{pmatrix} \tag{3.1}$$

where ∇I is image gradient. Then a multilocal representation H_ρ of the normalized Hessian matrix H is constructed by making a double regularization step as follows:

$$H_\rho = G_\rho * \left[\frac{1}{\sqrt{1+\|\nabla I_\sigma\|^2}} \begin{pmatrix} \frac{\partial^2 I_\sigma}{\partial x^2} & \frac{\partial^2 I_\sigma}{\partial x \partial y} \\ \frac{\partial^2 I_\sigma}{\partial y \partial x} & \frac{\partial^2 I_\sigma}{\partial y^2} \end{pmatrix} \right] \tag{3.2}$$

where I_σ is Gaussian-smoothed version of I and σ is standard derivation of the Gaussian filter; G_ρ is a Gaussian kernel with standard derivation equal to ρ for regularizing the tensor field.

By denoting eigenvalues of the multilocal normalized Hessian matrix H_ρ as κ_1 and κ_2, we define the following two descriptors

$$\mu_r = \begin{cases} \frac{\tilde{\kappa}_1 - \tilde{\kappa}_2}{\tilde{\kappa}_1 + \tilde{\kappa}_2} & \text{if } \kappa_1 < 0 \\ 0 & \text{if } \kappa_1 \geq 0 \end{cases}, \tag{3.3}$$

$$\mu_v = \begin{cases} \frac{\tilde{\kappa}_1 - \tilde{\kappa}_2}{\tilde{\kappa}_1 + \tilde{\kappa}_2} & \text{if } \kappa_1 > 0 \\ 0 & \text{if } \kappa_1 \leq 0 \end{cases}, \tag{3.4}$$

where $\tilde{\kappa}_1 = \max(|\kappa_1|, |\kappa_2|)$ and $\tilde{\kappa}_2 = \min(|\kappa_1|, |\kappa_2|)$, to detect ridges and valleys, respectively. It's worth mentioning that μ_r and μ_v reach their highest value of 1 in the presence of ridges and valleys, respectively, as described in [56, 57, 58, 59].

We denote v_1 and v_2 as the eigen-directions corresponding to the greatest and lowest eigenvalue in absolute value, respectively. v_1 and v_2 can also be interpreted as the directions across and along the local crease, respectively. The *diffusion tensor* D is constructed to have the same eigenvectors v_1 and v_2 as Hessian matrix H_ρ. The eigenvalues of D are chosen as

$$\begin{aligned} \lambda_1 &= \epsilon & \epsilon &\in (0,1) \\ \lambda_2 &= \alpha\mu_r + \beta\mu_v & \alpha, \beta &\in [0,1]\,, \end{aligned} \tag{3.5}$$

where α and β control the diffusion strength in the local crease direction in the presence of ridges and valleys, respectively; the value ϵ controls the diffusion strength across the crease. The positive value of ϵ ensures the semidefinite property of the diffusion tensor D.

Using creaseness diffusion tensor D, we filter the terrain image through the following diffusion equation.

$$\frac{\partial I}{\partial t} = \text{div}(D \cdot \nabla I) \tag{3.6}$$

We employ the discrete method (convolution in discrete domain) as described in [60] to solve the continuous diffusion equation.

3.2.2 MLSEC Operator

The creaseness measure we employ for crease extraction is known as a Multilocal Level-Set Extrinsic Curvature (MLSEC) developed in [61]. This is inspired from medical imaging as it is employed in arterial imaging. We have modified this approach to work with drainage patterns because the model of randomness exhibited by human circulatory system is in parallel to that of the drainage patterns in nature. Creases constructed by MLSEC are invariant with respect to spatial translation, spatial rotations, and uniform spatial magnification, and contain fewer crease discontinuities. Here, we consider the case in 2D ($d = 2$) with $\mathcal{B}$ composed of the eight nearest neighbors of each pixel ($r = 8$). That is, for the pixel $P_{i,j}$ of coordinates $[i, j]$, we have $\mathcal{B} = \{P_{i,j-1}, P_{i+1,j-1}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1}, P_{i-1,j+1}, P_{i-1,j}, P_{i-1,j-1}\}$ and $\mathcal{N} = \{\mathbf{n}_N, \mathbf{n}_{NE}, \mathbf{n}_E, \mathbf{n}_{ES}, \mathbf{n}_S, \mathbf{n}_{SW}, \mathbf{n}_W, \mathbf{n}_{WN}\}$ as shown in Fig. 3.2.

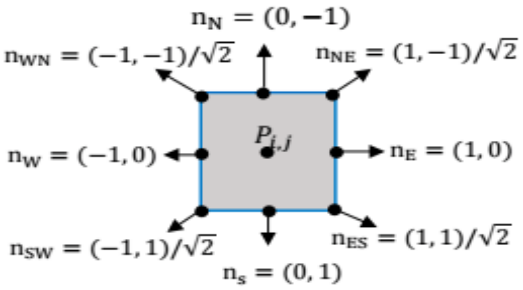


Figure 3.2: 8-adjacency neighborhood of P_{ij} and their unit normal for MLSEC computation

3.3 Minutiae Extraction

Although nine types of minutiae are introduced in Section 3.1, we primarily focus on crease endings and bifurcations, which have been proved to be the most distinguishing in [63, 64]. We employ the Crossing Number (CN) method from [65] to extract crease endings and bifurcations from the skeleton crease images. For each crease pixel P, this approach scans its 8 neighbors in anti-clockwise direction as shown in Fig. 3.3, and computes the intensity difference in absolute value between two adjacent neighbors. The CN value of a crease pixel P is then defined as the sum of intensity differences divided by 2, as given in the following equation:

$$CN = 0.5 \sum_{n=1}^{8} |I(P_n) - I(P_{n+1})|, \quad P_9 = P_1 \tag{3.7}$$

where $I(P_n)$ represents the intensity value of pixel P_n. Intensity values of a crease and non-crease pixel in the crease image are equal to 0 and 1, respectively.

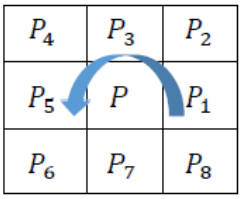


Figure 3.3: 8-adjacency neighborhood of P for CN computation

For this formula, if $CN = 1$, the pixel corresponds to the crease ending, and if $CN = 3$, the pixel corresponds to a bifurcation, as shown in Fig. 3.4. In this figure, blue and white squares represent crease and non-crease pixels, respectively.

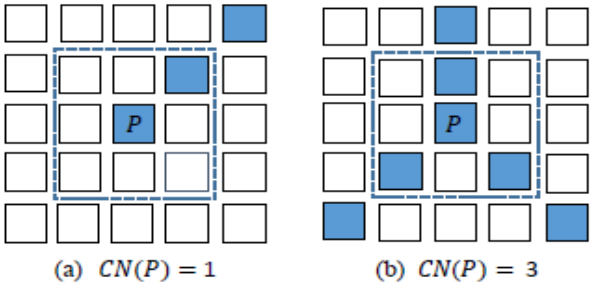


Figure 3.4: CN value of a creasing ending or bifurcation pixel.

In our research work, we represent a minutiae using quaternary structure. Each minutiae m_k is described as $m_k = \{x_k, y_k, \theta_k, T_k\}$, where x_k and y_k represent the normalized x and y coordinates of the minutiae, respectively; θ_k represents the angular direction of the crease associated with the minutiae as shown in Fig. 3.5; and T_k refers to the type of minutiae (e.g., endings or bifurcations). Once we identify minutiae m_k using CN operator, we compute its coordinates x_k and y_k, as well as orientation θ_k, as described in Section 3.3.1 and 3.3.2.

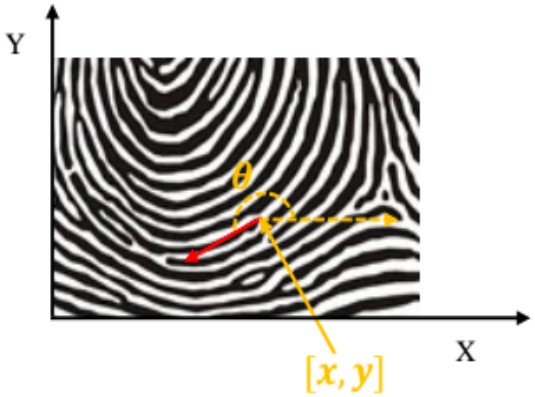


Figure 3.5: One example of crease bifurcation, where the orientation value θ is equal to the angular direction of the crease from the East.

3.3.1 X-Y Coordinate

Suppose pixel P of coordinate $[i, j]$ on 2D image plane is identified as minutiae m_k, then X-Y coordinate (x_k, y_k) of minutiae m_k is computed as follows:

$$x_k = j/W, \tag{3.8}$$

$$y_k = (H + 1 - i)/H, \tag{3.9}$$

where W and H represent width and height of the raw terrain image, respectively. It's easy to make the observation that $x_k \in [0, 1]$ and $y_k \in [0, 1]$.

3.3.2 Orientation

Taking advantage of analogy between human fingerpints and drainage patterns, we employ the pixel-wise scheme as described in [67] to estimate the local orientation of creases contained

in the terrain image. This method is attractive as it produces a finer and more accurate estimation of the orientation field compared to other block-wise approach as described in [66]. The steps for calculating the orientation value $\theta(i,j)$ at pixel $P=(i,j)$ are described below.

1. Identify the block of size $W \times W$ centered at pixel (i,j) in the crease image. For each pixel (u,v) in the block, compute the graidents in both X and Y directions, denoted as $\partial_x(u.v)$ and $\partial_y(u,v)$, respectively. Here, Sobel operators G_x and G_y are employed to compute $\partial_x(u.v)$ and $\partial_y(u,v)$, respectively.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.10}$$

2. Employ least mean square estimation approach to compute the local orientation $\theta_e(i,j)$ of pixel P using the following equations.

$$V_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2\partial_x(u,v)\partial_y(u,v) \tag{3.11}$$

$$V_y(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} \partial_x^2(u,v)\partial_y^2(u,v) \tag{3.12}$$

$$\theta_e(i,j) = \frac{1}{2}\tan^{-1}\frac{V_y(i,j)}{V_x(i,j)} \tag{3.13}$$

3. Smooth the orientation field in a local neighborhood by converting the orientation field into a continuous vector field. Gaussian smoothing is performed using the following equation:

$$\Phi'_x(i,j) = \sum_{u=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} G(u,v)\Phi_x(i-u,j-v), \tag{3.14}$$

$$\Phi'_y(i,j) = \sum_{u=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} G(u,v)\Phi_y(i-u,j-v), \tag{3.15}$$

where

$$\Phi_x(i,j) = \cos(2\theta_e(i,j)), \tag{3.16}$$

$$\Phi_y(i,j) = \sin(2\theta_e(i,j)). \tag{3.17}$$

Here, G is a Gaussian filter of size $w_\Phi \times w_\Phi$; $\Phi_x(i,j)$ and $\Phi_y(i,j)$ represent the x and y components of the orientation field, respectively.

4. The final smoothed orientation value O at pixel P is defined in Eq. 3.18.

$$O(i,j) = \frac{1}{2}\tan^{-1}\frac{\Phi'_y(i,j)}{\Phi'_x(i,j)} \tag{3.18}$$

Once orientation field O is computed, orientation θ_k of minutiae m_k, located at row i and column j is defined in Eq 3.19.

$$\theta_k = O(i,j) \tag{3.19}$$

3.4 Experiments

We operate MLSEC-based drainage pattern extraction method as well as CN-based minutiae extraction approach on both aerial and GPU-rendered terrain image in order to test effectiveness of these two approaches.

3.4.1 MLSEC-based Drainage Pattern Extraction

As map-aided UAS navigation involve matching aerial images against terrain landmarks from GPU render. We experiment MLSEC-based crease extraction approach on both aerial and GPU-rendered terrain images to show its effectiveness. For a terrain, ridges and valleys are dual in the sense that valleys of a terrain image are the ridges of the inverted image. In our work, we focus on terrain valleys. Parameter values for diffusion filtering and crease orientation estimation are given in Table 3.1. It is worth mentioning that values of parameter K, σ and w_1 depend on spatial resolution of terrain images. σ and w_1 control the extent of initial image smoothing in order to obtain robust derivatives. The higher resolution of the terrain image, the larger values of σ and w_1 are to take more neighboring pixel information into consideration. K control diffusion strength. The larger value of K, the sharper crease features we obtain. In the following experiment, each pixel corresponds to one terrain cell of 10m by 10m. Under

this scenario, we experimentally set K as 5 based on the observation that further increase in K could not help improve performance. Besides, we set σ and w_1 as 1 and 5, respectively, in order to smooth the terrain image while keeping minor drainage patterns. Based on discussion in [55], we set ϵ, α and β as 0.001, 1, and 1, respectively for ridge-valley diffusion.

Table 3.1: Parameter values for terrain valley extraction

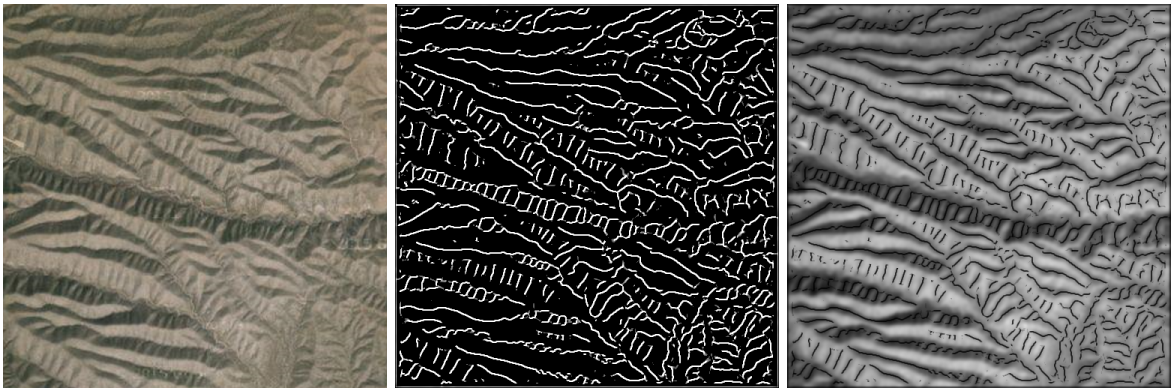
Paramter	**Description**	**Relevance**	**Value**
K	Total iteration number for diffusion filtering	Diffusion process	5
σ	Standard derivation of Gaussian filter used to smooth original mountain images	Diffusion process	1
w_1	Size of Gaussian filter used to smooth original mountain images	Diffusion process	5
ρ	Standard derivation of Gaussian kernel used to regularize the tensor field	Diffusion process	1
w_2	Size of Gaussian kernel used to regularize the tensor field	Diffusion process	3
ϵ	Parameter controlling the diffusion strength in the direction across the creases	Diffusion process	0.001
α	Parameter controlling the diffusion strength along creases in presence of ridges	Diffusion process	1
β	Parameter controlling the diffusion strength along creases in presence of valleys	Diffusion process	1
W	Block Size	Orientation Estimation	16
σ_s	Standard deviation of Gaussian filter used to smooth orientation field	Orientation Estimation	3
w_Φ	Size of Gaussian filter used to smooth orientation field	Orientation Estimation	$6 * \sigma_s = 18$

We experiment with MLSEC based crease extraction approach over Hualapai Peak area ($35°03'59''N, 113°45'58''W$) in Arizona to extract terrain valleys. Terrain images of Hualapai Peak from monocular camera and GPU render are shown in Fig. 3.6a and Fig. 3.6d, respectively. Size of both images are equal to 512 by 512 pixels, where each pixel corresponds to one terrain cell of 10m by 10m. We apply the crease extraction approach on these two images, and display the extracted terrain valleys from aerial and rendered images in Fig. 3.6b and Fig. 3.6e, respectively. To facilitate observation, we overlap extracted valleys from both aerial and GPU-rendered images with the greyscale terrain image, and present the results in Fig. 3.6c and Fig. 3.6f, respectively. In these two figures, black pixels represent extracted terrain valleys. We make the following observations from experimental results:

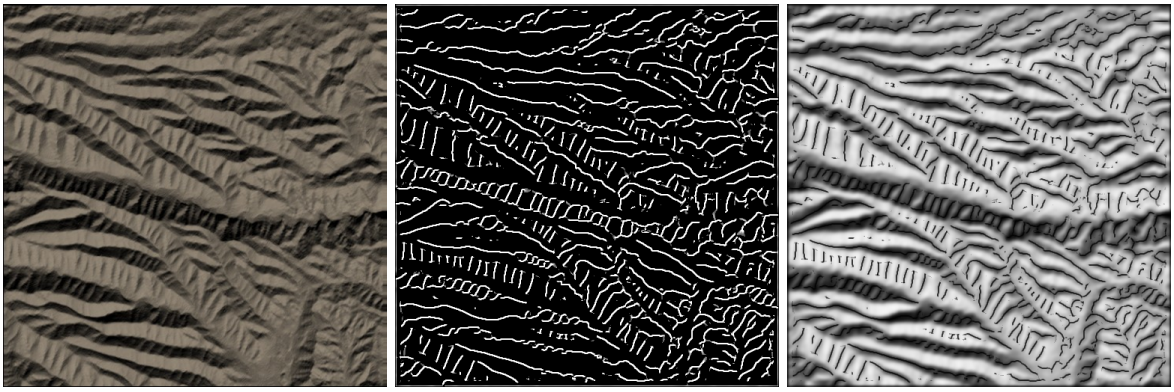
- By observing Fig. 3.6c and Fig. 3.6f, we notice that MLSEC-based crease extraction

approach has the capability to accurately locate drainage patterns in terrain images from both monocular camera and GPU render, although with some undesirable interruptions.

- By comparing Fig. 3.6b with Fig.3.6e, we notice that terrain valleys from aerial and GPU-rendered images of the same area are almost the same in terms of spatial distribution, although with slight differences in details (i.e., introducing interruptions). This observation validates the effectiveness of our terrain rendering process. In addition, the slight difference can be explained by the noise introduced in the rendering process.



(a) Real-time terrain image (b) Terrain valley in aerial image (c) Minutiae likelihood of imager data

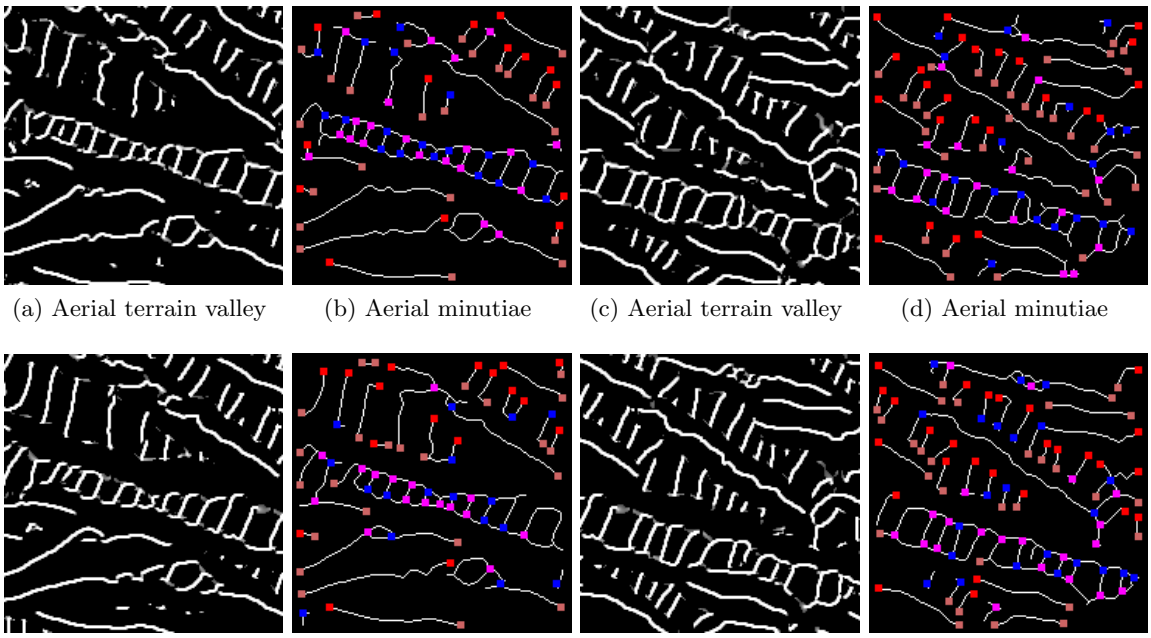


(d) GPU rendering of GRRR (e) Terrain valley in GPU render (f) Minutiae Likelihood of GRRR

Figure 3.6: Terrain valley extraction on Hualapai Peak area.

3.4.2 Minutiae Extraction

We employ CN-based minutiae extraction approach to obtain valley endings and bifurcations from valley images as shown in Fig. 3.6b and Fig. 3.6e. In detail, we first divide both valley images into 3 by 3 blocks, with size of each block equal to 180 by 180 pixels. We then randomly choose two valley blocks as samples for minutiae extraction.



(a) Aerial terrain valley (b) Aerial minutiae (c) Aerial terrain valley (d) Aerial minutiae

(e) Terrin valley of GRRR (f) Minutiae of GRRR (g) Terrin valley of GRRR (h) Minutiae of GRRR

Figure 3.7: Valley ending and bifurcation extraction on Hualapai Peak area

Two valley blocks from imager data and GRRR data are shown in Fig. 3.7a and Fig. 3.7c, and Fig. 3.7e and Fig. 3.7g, respectively. It is worth mentioning valleys as shown in Fig. 3.7a and Fig. 3.7e are from the same terrain block, but from different sources (i.e., one from camera and the other from GPU render). Similarly, valleys as shown in Fig. 3.7c and Fig. 3.7g are from the same terrain block, but from different sources. We operate minutiae extraction approach on four valley blocks as shown in Fig. 3.7a, Fig. 3.7c, Fig. 3.7e and Fig. 3.7g, and present identified minutiae in Fig. 3.7b, Fig. 3.7d, Fig. 3.7f, and Fig. 3.7h, respectively. In these four minutiae figures, red and orange squares are used to represent valley endings. Two colors are introduced to differentiate valley endings with different orientations(red for $[0, 180°]$, and

orange for $(180, 360°)$). Similarly, blue and red purple squares are used to represent valley bifurcation in different orientation range (blue for $[0, 180°]$, and purple red for $(180, 360°)$). By comparing these four minutiae images, we make the following observations:

- Spatial distribution of both minutiae and valleys in image pair of Fig. 3.7b and Fig. 3.7f, as well as image pair of Fig. 3.7d and Fig. 3.7h are similar, although with slight differences.
- Spatial distribution of both minutiae and valleys in image pair of Fig. 3.7b and Fig. 3.7h, as well as image pair of Fig. 3.7d and Fig. 3.7f are quite different.

The first observation further validates the accuracy of our terrain rendering approach based on displacement mapping. The second observation corresponds to our expectation that spatial distribution of both minutiae and valleys in different terrain blocks differ from each other. In future work, we aim to develop shape descriptors taking advantage of minutiae as well as crease distribution to recognize terrains.

3.5 Conclusions

In this chapter, we employ the MLSEC-based crease extraction technique, followed by CN based minutiae extraction, to obtain valley endings and bifurcations in terrain areas. Based on the above experiment results as well as the following experiment results as shown in Chapter 4, we observe that that spatial distribution of both minutiae and valleys from the same area, but different imaging source, are similar, although with sight difference. However, spatial distribution of both minutiae and valleys from different areas are quite different. Based on experimental observation we conclude that spatial distribution of both minutiae and crease is an attractive feature for terrain recognition purpose.

CHAPTER 4. MINUTIAE BASED LOCATING IN TERRAIN

In GPS challenged terrain environments, UASs utilize information from other inertial and air data sensor for navigation. These sensor equipages integrate PVA state over time resulting in cumulative measurement errors. That means PVA state from inertial and air data sensor just represents approximate location of UASs. Accurate position is bounded by maximum possible state drift. This bounded flight area within which accurate position of UAS is located is called active flight region in our research work.

Our UAS navigation augmentation system aims to locate accurate position within active flight region via terrain matching. Main idea is matching aerial images to landmarks within active flight region in order to find a match. Landmarks are defined as fixed-size terrain areas with strong drainage patterns. Image size of each landmark is set to be the same as that of aerial images. We give an example as shown in Fig. 4.1 to illustrate the locating process. Suppose GPS is lost due to signal blockage when UAS is operating over Kings Peak area in Utah. At a given time, UAS requires accurate position information for navigation. To achieve this goal, our system first identifies active flight region based on PVA state from available sensor data as well as estimated maximum possible state drift. Active flight region is represented by the yellow circle in Fig. 4.1. It then fetches GRRR data of landmarks within active flight region from memory, and generates life-like appearances of these landmarks using weather and time information from sensor data. Our system matches the aerial image from monocular camera to each landmark in order to find a match. In Fig. 4.1, blue and black rectangles represent pre-defined landmarks and current position of UAS, respectively.

Mathematical description of minutiae-based matching between an aerial image AI and a GPU-rendered landmark GI is given as follows. Let A and G be minutiae set from aerial image

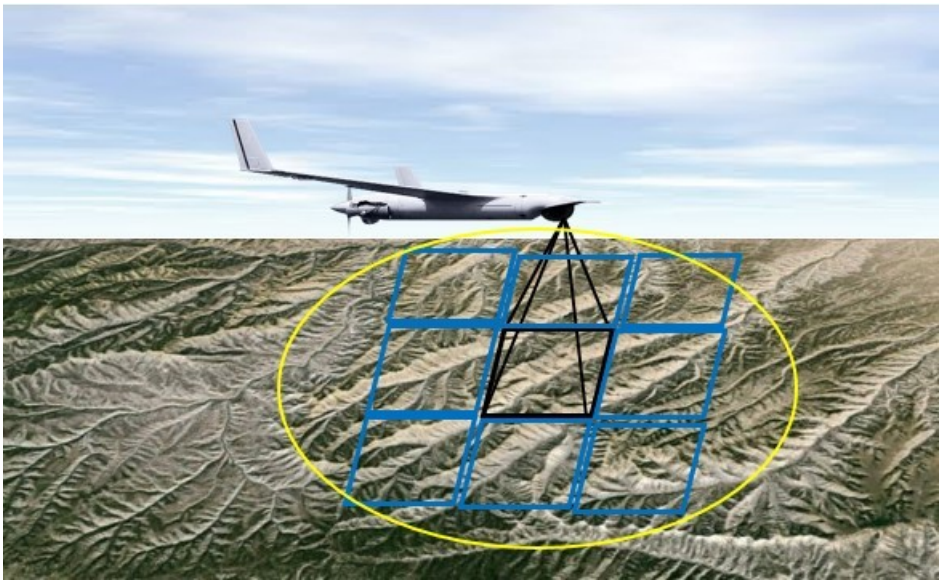


Figure 4.1: The terrain environment shows Kings Peak in Utah. Yellow circle, black rectangle and blue rectangles represents active flight region, aerial image and terrain landmarks, respectively.

AI and landmark GI, respectively:

$$A = \{m_{A_1}, m_{A_2}, \cdots, m_{A_p}\}, \ \textbf{where}\ m_{A_i} = \{x_{A_i}, y_{A_i}, \theta_{A_i}\},\ 1 \leq i \leq p \tag{4.1}$$

$$G = \{m_{G_1}, m_{G_2}, \cdots, m_{G_q}\}, \ \textbf{where}\ m_{G_j} = \{x_{G_j}, y_{G_j}, \theta_{G_j}\},\ 1 \leq j \leq q \tag{4.2}$$

where each minutiae m_k is represented as a triplet $m_k = \{x_k, y_k, \theta_k\}$ with x_k and y_k representing the normalized X-Y coordinates of the minutiae, and θ_k the orientation of the minutiae. It is worth mentioning that as we do not consider the type information for terrain matching, we drop the minutiae type in minutiae stracture. For terrain matching purpose, we aim to find the best matching minutiae $m_{G_j} \in G$ for each minutiae $m_{A_i} \in A$. To achieve this goal, we first develop two shape descriptors to measure similarity between two minutiae, and then design a matching process to identity minutiae pairings from two input minutiae sets A and G.

4.1 Shape Context Descriptor

In Section 3.4.2, we make the observation from experiment results that the more alike two terrain blocks look, the more similar minutiae and crease distribution are in these two blocks. Based on the observation, we develop two shape context descriptors, including both minutiae and crease shape descriptors, to measure neighborhood similarity between two different minutiae. Minutiae and crease shape descriptors are designed to provide localized spatial survey of minutiae and creases distribution around a given minutiae, respectively.

4.1.1 Minutiae Shape Descriptor

Taking advantage of the analogy between human fingerprint and terrain valleys, we modify Enhanced Shape Descriptor as described in [62] to work with our terrain matching. For each minutiae $m_{A_i} \in A$, we build its minutiae shape descriptor as a histogram h_{A_i} involving the remaining $p-1$ minutiae in A as shown in Eq. 4.3.

$$h_{A_i}(k) = \#\{A_e \neq A_i : (x_{A_e} - x_{A_i}, y_{A_e} - y_{A_i}) \in bin(k)\} \tag{4.3}$$

We divide the spatial geometric regions to be uniform in Euclidean space as shown in Fig. 4.2. The uniform division is chosen to ensure that areas of bins closer to the reference minutie m_{A_i} are smaller than that of distant bins, assigning larger weight to minutiae closer to m_{A_i}. Height $h_{A_i}(k)$, $k \in \{1, 2, \cdots, K\}$ of each bin is therefore computed as the tally of minutiae m_{A_e}, with distance d and direction θ relative to m_{A_i} satisfying the following requirement:

$$r_k^l \leq d < r_k^h \quad \text{and} \quad \theta_k^l \leq \theta < \theta_k^h, \tag{4.4}$$

where

$$d = \sqrt{(x_{A_e} - x_{A_i})^2 + (y_{A_e} - y_{A_i})^2}, \tag{4.5}$$

$$\theta = \arctan(\frac{y_{A_e} - y_{A_i}}{x_{A_e} - x_{A_i}}), \tag{4.6}$$

$$r_k^l = (1/N_d) * \mathbf{floor}((k-1)/N_\theta) \quad \text{and} \quad r_k^h = (1/N_d) * (\mathbf{floor}((k-1)/N_\theta) + 1), \tag{4.7}$$

$$\theta_k^l = (2 * Pi/N_\theta) * ((k-1)\%N_\theta) \quad \text{and} \quad \theta_k^h = (2 * Pi/N_\theta) * ((k-1)\%N_\theta + 1). \tag{4.8}$$

Here, **floor**(x) rounds x to the nearest integer less than or equal to x. N_d and N_θ represent bin numbers for distance d and direction θ, respectively. K denotes the total number of bins, whose value is equal to $N_d \times N_\theta$. The minutiae shape descriptor is then constructed for each minutiae $m_{A_i} \in A$, and likewise, each $m_{G_j} \in G$, providing a localized spatial survey of minutiae distribution for each terrain image.

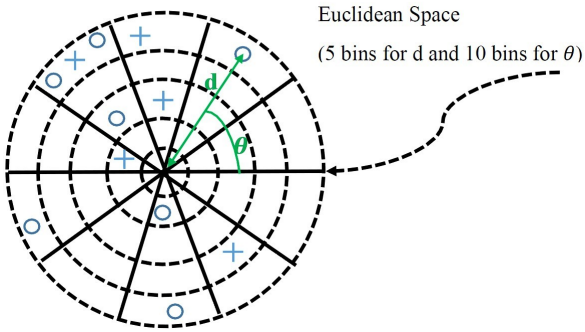


Figure 4.2: Histogram bins used to create minutiae shape descriptor for a given minutiae point. Crease bifurcations and endings are denoted by '+' and 'o', respectively.

Let $MCost_{ij} = MCost(A_i, G_j)$ denote the cost of matching minutiae $m_{A_i} \in A$ to minutiae $m_{G_j} \in G$ based on minutiae shape descriptor. Considering that the minutiae shape descriptors are distributed as histograms, we employ χ^2 statistic as described in [68] to compute $MCost_{ij}$ as given in Eq. 4.9. In our application, minutiae number in each bin varies. The height difference between small bins should be more important than the difference between large bins. We therefore choose chi-square distance measure to take this into account.

$$MCost_{ij} = MCost(A_i, G_j) = \sum_{k=1}^{K} \frac{(h_{A_i}(k) - h_{G_j}(k))^2}{h_{A_i}(k) + h_{G_j}(k)} \tag{4.9}$$

where h_{A_i} and h_{G_j} represent the K-bin histogram of minutiae m_{A_i} and m_{G_j}, respectively. We make the observation that the more similar between two descriptors h_{A_i} and h_{G_j}, the smaller cost $MCost_{ij}$ is in order to match minutiae m_{A_i} with m_{G_j}.

4.1.2 Crease Shape Descriptor

Crease shape descriptor is designed to provide local samples of crease presence around a given minutiae in a concentric layout, as shown in Fig. 4.3. For a given minutiae $m_{A_i} \in A$, vlaue of its s^{th} sample on the c^{th} concentric circle is calculated as

$$S_{A_i}(c,s) = \begin{cases} 1 & \text{if } \sum_{u=u_{A_i}^{c,s}-1}^{u_{A_i}^{c,s}+1} \sum_{v=v_{A_i}^{c,s}-1}^{v_{A_i}^{c,s}+1} AV(u,v) > 0 \\ 0 & \text{else} \end{cases} \tag{4.10}$$

with

$$u_{A_i}^{c,s} = \mathbf{floor}(u_{A_i} + r_c * cos(\theta_s)), \tag{4.11}$$

$$v_{A_i}^{c,s} = \mathbf{floor}(v_{A_i} + r_c * sin(\theta_s)), \tag{4.12}$$

where $AV(u,v)$ represents the intensity value at pixel (u,v) in binary valley image AV; r_c and θ_s are distance and direction of the sample point relative to minutiae m_{A_i}, respectively; (u_{A_i}, v_{A_i}) and $(u_{A_i}^{c,s}, v_{A_i}^{c,s})$ represent column-row values of minutiae m_{A_i} and the sample point, respectively. In Eq. 4.10, $S_{A_i}(c,d) = 1$ means that there exist one crease around the sample point in AV.

In our work, we consider concentric circles, which fit inside each other and are the same distance apart all the way around. Number of sample points on each circle is fixed, so that more samples within a unit arc length are considered for circles closer to the reference minutiae m_{A_i}. Distance r_c and direction θ_s are therefore computed as follows:

$$r_c = \mathbf{min}\{H, W\} * c/N_c, \quad c \in \{1, 2, \cdots, N_c\}, \tag{4.13}$$

$$\theta_s = 2 * Pi * s/N_s, \quad s \in \{1, 2, \cdots, N_s\}, \tag{4.14}$$

where H and W are height and width of aerial image AI, respectively; N_c and N_s represent numbers of concentric circles and samples on each circle, respectively. Crease shape descriptor S_{A_i} is then constructed for each minutiae $m_{A_i} \in A$, and likewise, each $m_{G_j} \in G$.

Let $CSim_{ij} = CSim(A_i, G_j)$ denote the crease similarity between minutiae $m_{A_i} \in A$ and $m_{G_j} \in G$ in their neighborhood. For one minutiae pair that are similar to each other, we expect large number of overlapping creases at sample points. We therefore define the crease similarity

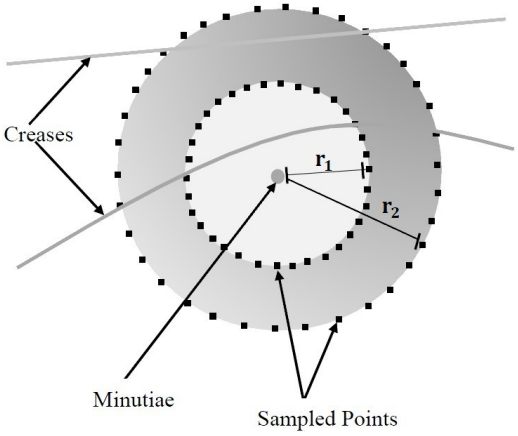


Figure 4.3: Concentric circles to construct local crease shape descriptor of a given minutiae.

$CSim_{ij}$ as ratio of number of overlapping creases to total number of creases over all sample points, which are described by the following equation:

$$CSim_{ij} = CSim(A_i, G_j) = \frac{2 * OS(A_i, G_j)}{TS_{A_i} + TS_{G_j}}, \tag{4.15}$$

with

$$OS(A_i, G_j) = \#\{(c, s) : S_{A_i}(c, s) = 1 \text{ and } S_{G_j}(c, s) = 1\}, \tag{4.16}$$

$$TS_{A_i} = \sum_{c=1}^{C} \sum_{s=1}^{S} S_{A_i}(c, s), \tag{4.17}$$

$$TS_{G_j} = \sum_{c=1}^{C} \sum_{s=1}^{S} S_{G_j}(c, s), \tag{4.18}$$

where $OS(A_i, G_j)$ is the total number of sample points where terrain creases appear for both minutiae m_{A_i} and m_{G_j}; TS_{A_i} and TS_{G_j} represent the total number of sample points where terrain creases appear for minutiae m_{A_i} and m_{G_j}, respectively. Value of $CSim_{ij}$ ranges from 0 to 1. The larger value of $CSim_{ij}$, the more similar between minutiae m_{A_i} and m_{G_j} in local crease distribution.

4.2 Registration

In GPS-challenged terrain environments, our system involves matching aerial images to landmarks in order to find a match. To generate a landmark, we fetch GRRR data from

memory and render the landmark by setting the camera to reflect the air sensor returned flight orientation. However, orientation of the same terrain block on 2D aerial image plane is actually determined by actual flight orientation of UASs. Deviation between actual and air sensor returned flight orientation results in translation and rotation between aerial and GPU rendered images of the same terrain block. As a result, although with both minutiae and crease shape descriptor available, we cannot proceed to find minutiae pairs from triplets without some pre-processing for the following two reasons:

- Aerial and GPU-rendered images of the same terrain block can differ in orientation, deeming the orientaion field in the triplet useless
- Aerial and GPU-rendered images of the same terrain block can differ in offset, deeming the X-Y fields in the triplet useless

4.2.1 Affine Transform

To address the issue of lacking invariant property in the triplet structure of minutiae, global registration is required to align aerial image AI and GPU-rendered landmark GI so that overlapping terrain regions have minimal geometric distance to each other. We employ affine transform as descibed in [69] to achieve the goal of registration. Using the minutiae set representation, each minutiae pair (m_{A_a}, m_{G_b}) defines one candidate affine transform T, which is calculated by using the orientation difference between minutiae m_{A_a} and m_{G_b} as the rotation component, and the X-Y coordinate difference as the translation component.

As the affine transform involving rotation and translation is invertible, applying affine transform to either minutiae set achieve the same registration performance. In our work, we apply the affine transform T to aerial minutiae set A, keeping minutiae set G from landmark fixed. Suppose (m_{A_a}, m_{G_b}) are chosen as reference minutiae pair for registration, minutiae triplet field of each minutiae $m_{A_i} \in A$ are updated according to Eq. 4.19 and Eq. 4.20.

$$\begin{bmatrix} x_{A_{T(i)}} \\ y_{A_{T(i)}} \end{bmatrix} = \begin{bmatrix} cos(\theta_\Delta) & -sin(\theta_\Delta) \\ sin(\theta_\Delta) & cos(\theta_\Delta) \end{bmatrix} * \begin{bmatrix} x_{A_i} \\ y_{A_i} \end{bmatrix} + \begin{bmatrix} x_\Delta \\ y_\Delta \end{bmatrix} \quad 1 \leq i \leq p, \tag{4.19}$$

$$\theta_{A_{T(i)}} = \theta_{A_i} - \theta_\Delta \quad 1 \leq i \leq p, \tag{4.20}$$

where θ_Δ is the orientation difference

$$\theta_\Delta = \theta_{A_a} - \theta_{G_b}, \tag{4.21}$$

and (x_Δ, y_Δ) is the location difference

$$x_\Delta = x_{G_b} - (cos(\theta_\Delta) * x_{A_a} - sin(\theta_\Delta) * y_{A_a}), \tag{4.22}$$

$$y_\Delta = y_{G_b} - (sin(\theta_\Delta) * x_{A_a} + cos(\theta_\Delta) * y_{A_a}), \tag{4.23}$$

between two reference minutiae $m_{A_a} \in A$ and $m_{G_b} \in G$, in order to super-impose aerial terrain image AI to GPU-rendered landmark GI with accurate overlap and uniform direction. New triplet field of minutiae m_{A_i} after applying affine transform T is denoted as $m_{A_{T(i)}} = \{x_{A_{T(i)}}, y_{A_{T(i)}}, \theta_{A_{T(i)}}\}$.

4.2.2 Reference Minutiae Pair Selection

For each candidate affine transform T, we define a minutiae pair set Φ_ψ. Here, ψ denotes the index pair reference for the transform T (i.e., (A_a, G_b) with $m_{A_{T(a)}} = m_{G_b}$). Each minutiae pair (m_{A_i}, m_{G_j}) in Φ_ψ satisfy the following two requirements: (1) it has each other as the closest minutiae from the opposite minutiae sets, (2) distance between two minutiae is less than a threshold r_δ after applying T. Mathematical description of Φ_ψ is given as follows:

$$\Phi_\psi = \{(m_{A_i}, m_{G_j}) \mid (m_{A_i}, m_{G_j}) \in \Phi_{\psi_A} \cap \Phi_{\psi_G} \text{ and } \sqrt{(x_{A_{T(i)}} - x_{G_j})^2 + (y_{A_{T(i)}} - y_{G_j})^2} < r_\delta\}, \tag{4.24}$$

with

$$\Phi_{\psi_A} = \{(m_{A_i}, m_{G_j}) \mid \arg\min_{i \in \{1,2,\cdots,p\}} \sqrt{(x_{A_{T(i)}} - x_{G_j})^2 + (y_{A_{T(i)}} - y_{G_j})^2}, \ 1 \leq j \leq q\}, \tag{4.25}$$

$$\Phi_{\psi_G} = \{(m_{A_i}, m_{G_j}) \mid \arg\min_{j \in \{1,2,\cdots,q\}} \sqrt{(x_{A_{T(i)}} - x_{G_j})^2 + (y_{A_{T(i)}} - y_{G_j})^2}, \ 1 \leq i \leq p\}, \tag{4.26}$$

where Φ_{ψ_A} contains all nearest minutiae pairs from A to each minutiae in G, and likewise, Φ_{ψ_G} contains all nearest minutiae pairs from G to each minutiae in A. In our work, each minutiae pair (m_{A_i}, m_{G_j}) in Φ_ψ is considered as a candidate minutiae correspondence due to closeness of their locations after applying affine transform T. More details about candidiate minutiae correspondence identification can be found in [64].

We then identify minutiae correspondences from the minutiae pair set Φ_ψ using both minutiae and crease shape descriptor. For each minutiae pairing, we expect similar minutiae as well as crease distribution in their neighborhood, and therefore develop the following minutiae paring selection criterion as described in Eq. 4.27.

$$\Phi_\psi^C = \{(m_{A_i}, m_{G_j}) |\ (m_{A_i}, m_{G_j}) \in \Phi_\psi \text{ and } SCost_{ij} < T_{scost} \text{ and } CSim_{ij} > T_{csim}\}, \tag{4.27}$$

where Φ_ψ^C refers to the set of minutiae correspondences; T_{scost} and T_{csim} represent the match cost threshold and crease similarity threshold, respectively, used to identify two minutiae from different sources as one minutiae pairing.

Let $n(\Phi_\psi^C)$ denote size of Φ_ψ^C, i.e., number of minutiae pairings. We expect the optimal reference minutiae pair ψ^* to maximize the total number of minutiae pairings. That is

$$\psi^* = \mathbf{arg\ max}_{\psi \in \Psi} n(\Phi_\psi^C) \tag{4.28}$$

where Ψ is the set of all possible reference minutiae pairs, and size of Ψ is equal to $p \times q$ for matching minutiae set A to G. Affine transform with ψ^* as reference minutiae pair is denoted as T^*. Additional pruning can be achieved by only allowing transforms where the reference minutiae pair ψ has a minutiae shape descriptor-based matching cost to meet

$$SCost_\psi < T_{scost} \tag{4.29}$$

and a crease similarity score to meet

$$CSim_\psi > T_{csim} \tag{4.30}$$

where T_{scost} and T_{csim} are empirically set. Such restrictions not only improve performance, but also speed up the registration process. Based on the above discussion, the correspondence identification process is detailed in Algorithm 1.

Algorithm 1: Minutiae Pairing Identification Process

Input: Minutiae set A from aerial image AI: $A = \{m_{A_1}, m_{A_2}, \cdots, m_{A_p}\}$, Minutiae set G from GPU-rendered image GI: $G = \{m_{G_1}, m_{G_2}, \cdots, m_{G_q}\}$, r_δ, T_{scost}, and T_{csim}

Output: Number of minutiae correspondences: N^c

```
// Initialization
```
$N^c_{pre} = 0, N^c = 0$, and $A' = A$;
```
// Loop each possible reference minutiae pair
```
for *i = 1 to p* **do**
  for *j = 1 to q* **do**
    $A = A'$;
    Step1: Compute the affine tranform T with m_{A_i} and m_{G_j} as reference minutiae pair based on Eq. 4.21, Eq. 4.22 and Eq. 4.23.
    Step2: For each minutiae $m_{A_i} \in A$, update its minutiae triplet field based on Eq. 4.19 and Eq. 4.20.
    Step3: Compute both minutiae and crease shape descriptors for minutiae $m_{A_{T(i)}}$ and minutiae m_{B_j} based on Eq. 4.3 and Eq. 4.10.
    Step4: Compute matching cost $SCost_{ij}$ as well as crease simularity score $CSim_{ij}$ based on Eq. 4.9 and Eq. 4.15, respectively; Compare $SCost_{ij}$ and $CSim_{ij}$ with given thresholds as follows.
    if $(SCost_{ij} < T_{scost}$ **and** $CSim_{ij} > T_{csim})$ **then**
      Go to Step 5
    else
      Continue
    Step5: Compute set of candidate minutiae correspondences Φ_ψ using Eq. 4.24, Eq. 4.25 and Eq. 4.26.
    Step6: Identify set of minutiae pairings Φ^C_ψ based on Eq. 4.27.
    Step7: Compare the total number of minutiae correspondences from current reference minutiae pair with N^c_{pre}, and update value of N^c and N^c_{pre} as follows.
    if $(N(\Phi^C_\psi) > N^c_{pre})$ **then**
      $N^c_{pre} = N(\Phi^C_\psi)$;
      $N^c = N(\Phi^C_\psi)$;
    else
      Continue
Return N^c
```
```

4.2.3 Terrain Similarity

Considering the varying minutiae density over different terrain areas, we conclude that it is not appropriate to measure similarity between A and G directly based on the total number of minutiae pairings. In our work, we define similarity score $TSimi(A, G)$ between A and G as the ratio of number of minutiae pairing to total number of minutiae in overlapping area. With ψ^* and $\Phi^C_{\psi^*}$ available, we calculate the value of $TSimi(A, G)$ using the following equations:

$$TSimi(A, G) = \frac{2 * n(\Phi^C_{\psi^*})}{op + oq}, \tag{4.31}$$

with

$$op = \#\{i \mid m_{A_{T^*(i)}} \text{ falls within } Overlap^{T^*}_{AG}\}, \tag{4.32}$$

$$oq = \#\{j \mid m_{G(j)} \text{ falls within } Overlap^{T^*}_{AG}\}, \tag{4.33}$$

where $Overlap^{T^*}_{AG}$ refers to the overlapping area between aerial image AI and GPU-rendered landmark GI after affine transform T^*; op and oq represent the total number of minutiae in A and G, which fall within the overlapping area, respectively. It's easy to make the observation that value of $TSimi(A, G)$ ranges from 0 to 1. The more similar between the two input minutiae set A and G, the higher value of $TSimi(A, G)$ we obtain.

4.3 Experiments

We operate minutiae-based matching algorithm, as described in Algorithm 1, to match terrain images from different areas. We provide two inputs to the matching algorithm, one from monocular camera and the other from GPU render. Outputs of the matching algorithm include the total number of minutiae pairings, as well as similarity score of two input terrain blocks as defined in Eq. 4.31. All experiments in this section are conducted at specific flight height such that each pixel on 2D image plane corresponds one terrain cell of size 10m $\times$ 10m. Size of each input image is equal to 180 by 180 pixels.

Before proceeding, we clarify values of parameters used for identifying minutiae pairings in Table 4.1. In Table 4.1, bin numbers N_d and N_θ for distance and direction are set to 5

and 10, respectively, so that there are sufficient bins to capture variation of minutiae desnsity under given image resolution and image size. N_c and N_s are set to 10 and 36, respectively, in order to reflect average crease density under given image resolution and image size. It is worth mentioning that we experimentally set values of T_{scost} and T_{csim} as a function of minutiae number in two input images. Concretely, we set looser requirement of T_{scost} and T_{csim} when minutiae numbers in input images are smaller. This can be explained by the fact that both minutiae and shape descriptors are more heavily affected by rendering noise when density of crease or minutiae in two inputs is smaller.

Table 4.1: Parameter values for identifying minutiae pairings

Paramter	**Description**	**Value**
N_d	Number of bins for distance	5
N_θ	Number of bins for direction	10
N_c	Number of concentric circles	10
N_s	Number of samples on each circle	36
r_δ	Distance threshold for anchor minutiae pairs	0.045
T_{scost}	Minutiae descriptor-based matching threshold	$0.30 * \frac{80}{\mathbf{min}(\mathbf{max}(\mathbf{p},\mathbf{q}),\ 80)}$
T_{csim}	Crease descriptor-based similarity threshold	$\mathbf{min}(0.35 * \frac{\mathbf{max}(\mathbf{p},\mathbf{q})}{80}, 0.45)$

4.3.1 Experiment 1

We run minutiae-based terrain matching algorithm on three pairs of images to show the performance of our algorithm. The first two refer to PVA matches, where two inputs, one from camera and the other from GPU render, cover the same terrain block. The last one is Non-PVA match, where field of view match with the area that the aircraft is not immediately flying over.

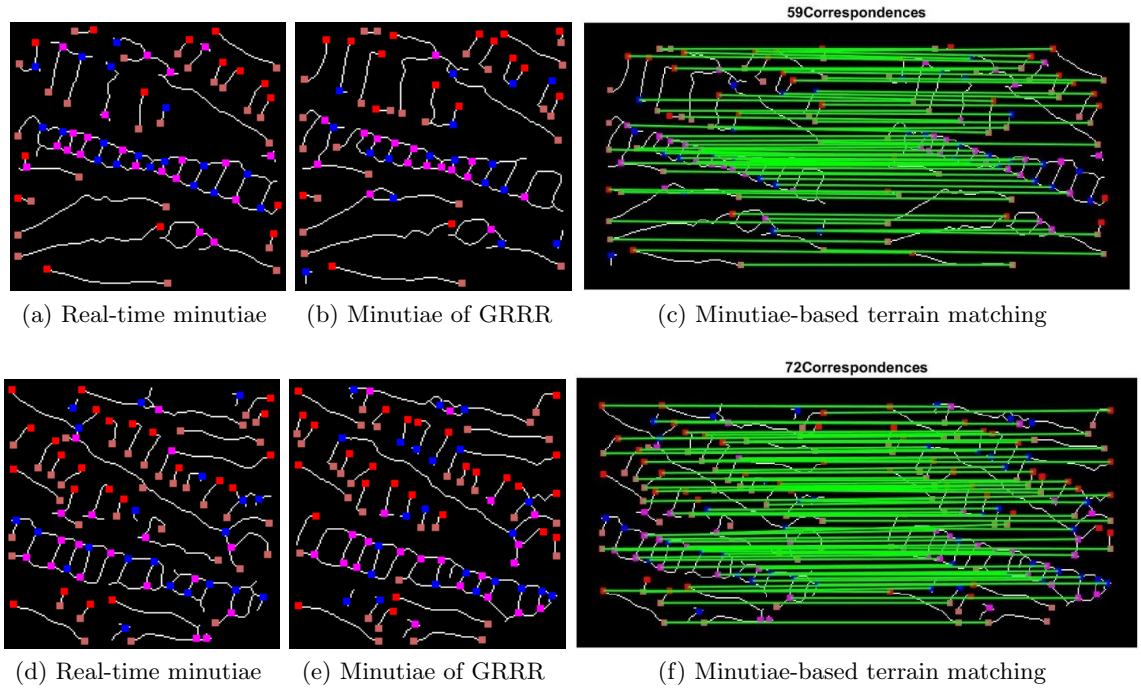
4.3.1.1 PVA Match

We run the minutiae-based matching algorithm on two pairs of valley images, including Fig. 3.7b and Fig. 3.7f, as well as Fig. 3.7d and Fig. 3.7h. For each pair, two valley images cover the same terrain block, but from different sources as described in Section 3.4.2.

For the first experiment, we input two minutiae sets as shown in Fig. 4.4a and Fig. 4.4b to our terrain matching algorithm. Outputs of the algorithm are presented in Fig. 4.4c. We

obtain a total of 59 minutiae pairings. As the total numbers of minutiae in these two input images are equal to 78 and 74, respectively, similarity score between Fig. 4.4a and Fig. 4.4b is calculated as 0.776. In the second PVA experiment, we input two minutiae sets as shown in Fig. 4.4d and Fig. 4.4e to our terrain matching algorithm. Outputs of the algorithm are presented in Fig. 4.4f. We obtain a total of 72 minutiae correspondences from the matching process. As the total numbers of minutiae in these two input images are equal to 97 and 101, respectively, similarity score between Fig. 4.4d and Fig. 4.4e is calculated as 0.727.

It is worth mentioning that we expect similarity score to be 1 for both experiments, as two inputs cover the same terrain block. But in fact, we get similarity scores of 0.776 and 0.727, respectively for the above two experiments. This can be explained by the noise in the rendering process due to differences in lighting conditions. Although not equal to 1, the matching results are still attractive as both similarity scores are higher than 0.70.



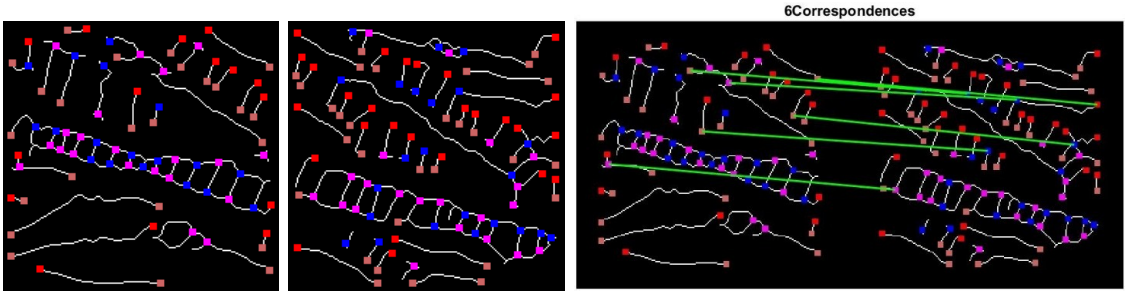
(a) Real-time minutiae (b) Minutiae of GRRR (c) Minutiae-based terrain matching

(d) Real-time minutiae (e) Minutiae of GRRR (f) Minutiae-based terrain matching

Figure 4.4: Two examples of PVA match over Hualapai Peak area

4.3.1.2 Non-PVA Match

In this experiment, we input two minutiae sets as shown in Fig. 4.5a and Fig. 4.5b to our terrain matching algorithm. Outputs of the algorithm are presented in Fig. 4.5c. We observe that we obtain a total of 6 minutiae pairings. As the total number of minutiae in two input images are equal to 78 and 101, respectively, similarity score between these two input terrain blocks is calculated as 0.067. The matching result is attractive as the similarity score of two different terrain blocks is small.



(a) Real-time minutiae (b) Minutiae of GRRR (c) Minutiae-based terrain matching

Figure 4.5: An example of Non-PVA match over Hualapai Peak area

4.3.1.3 Observations

We make the following observations from the above experiment results. Similarity scores of two PVA matches are higher than that of Non-PVA match. This observation corresponds to our expectation that the more similar two input terrain images look like, the higher similarity score we obtain with the minutiae-based matching algorithm. In the following, we use more sample data to further test our hypothesis as well as to find an appropriate matching threshold $TSimi$ for identifying PVA matches.

4.3.2 Experiment 2

We run two experiments on larger datasets to test effectiveness of drainage patterns in terrain recognition. Besides, we aim to find an appropriate matching threshold $TSimi$ to identify terrain matches based on experiment results.

4.3.2.1 Experiment in Utah

We operate the minutiae-based matching algorithm over terrain in Utah. We utilize images of 85 terrain blocks from different sections of a mission over Ridge Valley Mountain $(39°25'57''N, 110°11'24''W)$ as inputs of our experiment. Size of each image is equal to 180 by 180 pixels. Appearance of one mission section over Ridge Valley Mountain is shown in Fig. 4.6, which is of size 512 by 512 pixels. We input both the aerial imagery from on-board monocular camera and the GPU render of GRRR data of these sample terrain blocks to the algorithm to extract minutiae. We then choose a pair of them from different imaging sources as the inputs to the matching algorithm. Overall we have a total of $\binom{85}{2} + 85 = 3570 + 85 = 3655$ matching samples, among which 85 are PVA match, and the other 3570 are Non-PVA matches.

Figure 4.6: Appearance of one mission section over Ridge Valley Mountain.

- **Minutiae Numbers**

For each sample terrain block, number of minutiae in aerial imagery and GPU render of GRRR data might be different due to noise introduced by the rendering process. We therefore define

minutiae number in one sample terrain block as the average of minutiae numbers in aerial and GPU-rendered images. Numbers of minutiae in 85 terrain samples are depicted in Fig. 4.7. Average number of minutiae over these 85 terrain blocks is equal to 58.

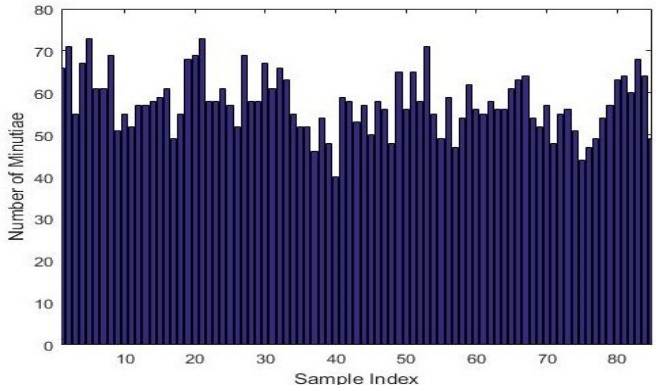
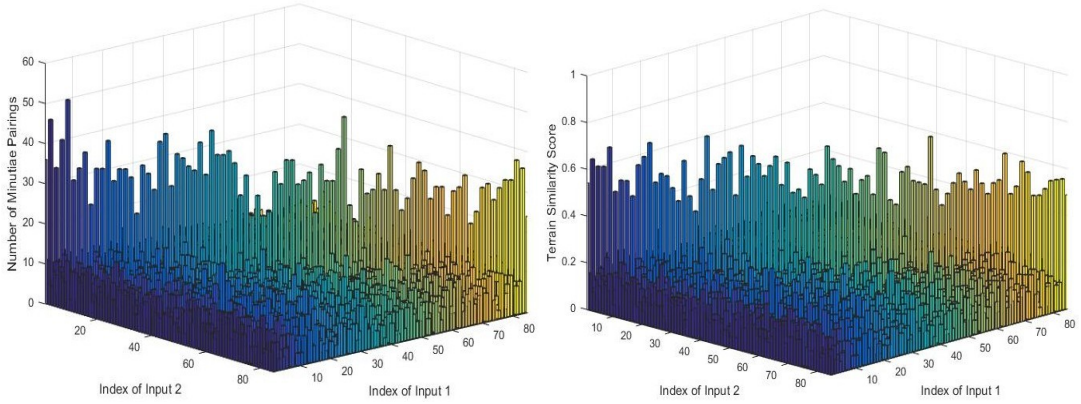


Figure 4.7: Numbers of minutiae in 85 terrain samples over Ridge Valley Mountain.

- **Outputs of Minutiae-based Terrain Matching**

We present the matching results, including both number of minutiae pairings as well as similarity score, between 85 terrain samples in Fig. 4.8. In both Fig. 4.8a and Fig. 4.8b, the diagonal represent PVA matches, and higher bars indicate higher certainty of matches. Off-diagonal represent Non-PVA matches where a field-of-view matches with an area that the aircraft is not immediately flying over.



(a) Number of minutiae pairings (b) Similarity score

Figure 4.8: Matching results between different terrain blocks over Ridge Valley Mountain

4.3.2.2 Experiment in Arizona

We run another minutiae-based terrain matching experiment over terrain in Arizona. We utilize images of 80 terrain blocks from different sections of a mission over Hualapai Peak ($35°03'59''N, 113°45'58''W$) as inputs of our experiment. Size of each image is equal to 180 by 180 pixels. Appearance of one mission section over Hualapai Peak is presented in Fig. 4.9, which is of size 512 by 512 pixels. We input both the aerial imagery from on-board monocular camera and the GPU render of GRRR data of these terrain samples to the algorithm to extract minutiae. We then choose a pair of them as the inputs to the minutiae-based matching algorithm. Overall we have a total of $\binom{80}{2} + 80 = 3160 + 80 = 3240$ matching samples, among which the 80 are PVA-match, and the other 3160 are Non-PVA matches.

Figure 4.9: Appearance of one mission section over Hualapai peak.

- **Minutiae Numbers**

Numbers of minutiae in 80 terrain blocks over Hualapai Peak are depicted in Fig. 4.10. Average minutiae number over these 80 terrain areas is equal to 80.

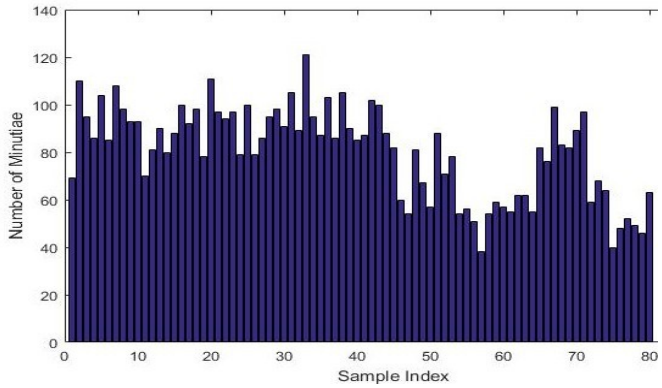
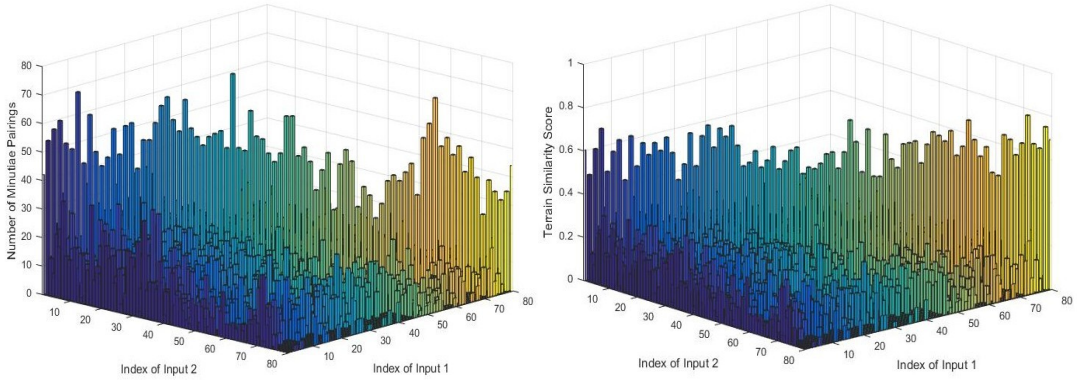


Figure 4.10: Numbers of minutiae in 80 terrain blocks over Hualapai Peak.

- **Outputs of Minutiae-based Terrain Matching**

We present outputs of the experiment, including both number of minutiae pairings as well as similarity score between 80 terrain samples, in Fig. 4.11. In both Fig. 4.11a and Fig. 4.11b, the diagonal represent PVA matches, and higher bars indicate higher certainty of matches. Off-diagonal represent Non-PVA matches where a field-of-view matches with an area that the aircraft is not immediately flying over.



(a) Number of minutiae pairings (b) Similarity score

Figure 4.11: Matching results between sample terrains over Hualapai Peak in Arizona

4.3.2.3 Performance Analysis

We describe two minutiae-based terrain matching experiments in detail in Section 4.3.2.1 and 4.3.2.2. In these two experiments, we have a total of $85 + 80 = 165$ PVA matches, and

$3570 + 3160 = 6730$ Non-PVA matches. To find an optimal threshold $TSimi$ to identify PVA matches, we build a frequency histogram for similarity scores of PVA match and Non-PVA match, respectively, in order to show distrution of similarity scores in these two scenarios.

We employ 165 samples on diagonal of both Fig. 4.8b and Fig. 4.11b to build frequency histogram of similarity scores from PVA matches. The histogram is presented in Fig. 4.12. In this plot, values in x-axis are similarity scores divided into 10 equally spaced bins betwen 0.4 and 0.9, with width of each bin equal to 0.05. Values in y-axis represent numbers of PVA matches in each bin. To facilitate observation, we also summarize counts of PVA matches in each bin in Table 4.2. Based on Table 4.2, we make the observation that similarity scores of 165 PVA matches are all larger than 0.40 in these two experiments. Especially, 96 percentage of similarity scores are between 0.45 and 0.75. Differences in similiarty scores among these 165 terrain blocks can be explained by variance of feature strength in these blocks. In detail, the stronger drainage pattern features in one terrain block, the higher similarity score we obtain from the PVA match.

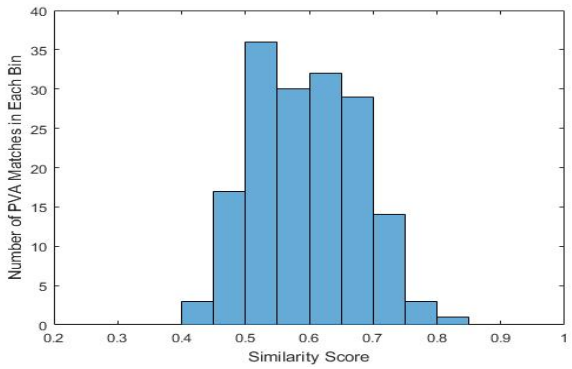


Figure 4.12: Frequency histogram of similarity score from PVA matches.

Table 4.2: Distribution of similarity scores from PVA matches

Bin Locations	$[0.40, 0.45)$	$[0.45, 0.50)$	$[0.50, 0.55)$	$[0.55, 0.60)$	$[0.60, 0.65)$	$[0.65, 0.70)$	$[0.70, 0.75)$	$[0.75, 0.80)$	$[0.80, 0.85)$
Counts	3	17	36	30	32	29	14	3	1
Frequency	0.0182	0.1030	0.2182	0.1818	0.1939	0.1758	0.0848	0.0182	0.0061

For Non-PVA match, we utilize 6730 off-diagonal similarity scores in both Fig. 4.8b and Fig. 4.11b as samples to build frequency histogram. The histogram is presented in Fig. 4.13. In this plot, values in x-axis are similarity scores divided into 8 equally spaced bins between 0 and 0.4. Values in y-axis represent numbers of Non-PVA matches in each bin. Similarily, to facilitate observation we also summarize counts of Non-PVA matches in each bin in Table 4.3. Based on Table 4.3, we make the observation that similarity scores of 6730 Non-PVA matches are all smaller than 0.35 in these two experiments. It is worth mentioning that a large majority of similarity scores are smaller than 0.30. Specifically, 99.79 percentage of similarity scores is smaller than 0.30. Only 0.21 percentage is between 0.30 and 0.35.

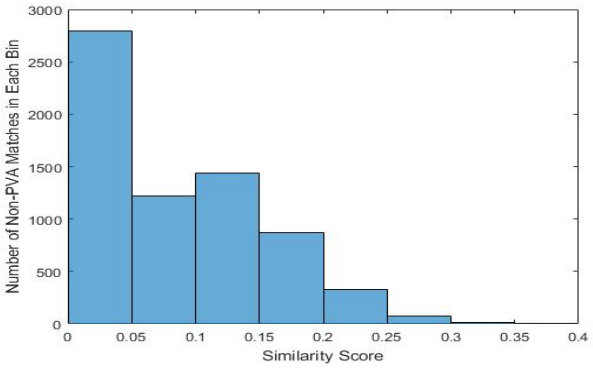


Figure 4.13: Frequency histogram of similarity score from Non-PVA matches.

Table 4.3: Distribution of similarity scores from Non-PVA matches

Bin Locations	$[0.00, 0.05)$	$[0.05, 0.10)$	$[0.10, 0.15)$	$[0.15, 0.20)$	$[0.20, 0.25)$	$[0.25, 0.30)$	$[0.30, 0.35)$
Counts	2793	1216	1436	871	329	71	14
Frequency	0.4150	0.1807	0.2134	0.1294	0.0489	0.0105	0.0021

Based on discussion, it's reasonable to set 0.35 as threshold to identify terrain matches based on similarity score. This threshold allows terrain blocks with relatively weaker drainage patterns as landmarks. By setting matching threshold $TSimi$ as 0.35, we obtain 0 misses

among 165 PVA matches, and 0 false positives among 6730 Non-PVA matches. It is worth mentioning that we run both experiments using CUDA by parallizing both crease/minutiae extraction and terrain matching algorithms. Experiment results prove that for each match, the whole terrain matching pipeline cost approximate 0.05s. More details can be found in [70]. Considering that speed of most UAS applications is around 30 knots [71], the minutiae-based matching algorithm has the capability to enable real-time navigation.

4.4 Conclusions

In this chapter, we first develop two shape descriptors, including both minutiae and crease, to describe neighborhood similarity between two minutiae. Using shape descriptors, we then design a matching process to identify minutiae pairings from two input minutiae sets. We finally propose a metric based on the number of minutiae pairings to identify terrain match. We operate the minutiae-based terrain matching algorithm on two datasets from different areas. Experiment results prove that our approach has the capability to recognize terrains.

CHAPTER 5. GRANULARITY OF DRAINAGE PATTERNS

In chapter 4, we operate minutiae-based terrain matching algorithm on terrain datasets from different areas. Experiment results prove the effectiveness of drainage patterns in terrain characterization. It is worth mentioning that all terrain images in the dataset are in fixed resolution. Each pixel corresponds to one terrain cell of size 10m × 10m. However, when UASs operate at different altitudes, resolution of images from on-board monocular camera varies. As a result, granularity of drainage patterns as well as number of minutiae in a fixed-size terrain image varies, i.e., the higher UASs operates, the more drainage patterns and minutiae in a fixed-size terrain image. In this chapter, we design two sets of experiments to investigate how flight altitude (i.e., minutiae number), affects performance of our minutiae-based terrain matching approach. For each set of experiments, we consider three scenarios that spatial resolutions of each pixel in fixed-size input images are equal to 10m × 10m, 5m × 5m , and 2.5m × 2.5m, respectively. In the following, we discuss each scenario in detail.

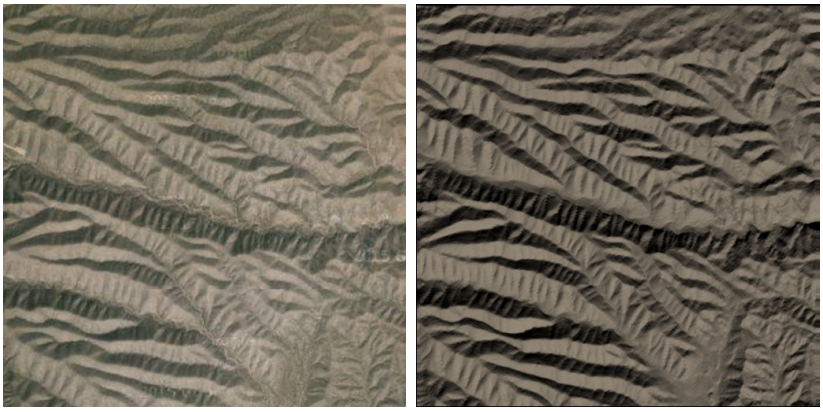
5.1 10m Resolution

We conduct two experiments to operate minutiae-based terrain matching algorithms over Hualapai Peak ($35°03'59''N, 113°45'58''W$) and Kings Peak ($31°22'58''N, 111°01'09''W$) in Arizona, respectively. Both experiments are conducted at the flight altitude such that each pixel on 2D image plane corresponds to one terrain cell of size 10m × 10m.

5.1.1 Hualapai Peak

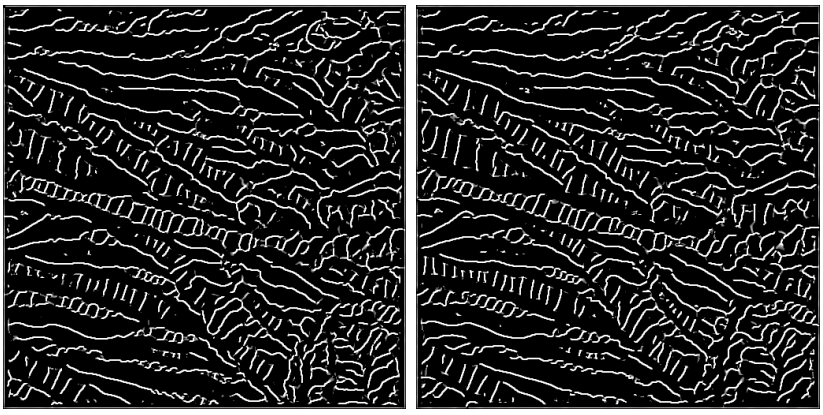
Fig. 5.1 shows the terrain area where we conduct our experiment. We utilize a down-looking fixed focus monocular camera to obtain aerial image of the terrain, which is presented in Fig. 5.1a. Size of the aerial image is equal to 512 by 512 pixels. We generate life-like

appearance of the same terrain area using GRRR data via displacement mapping. The output of GPU render is presented in Fig. 5.1b. To facilitate observation of crease and minutiae distribution in this area, we input both aerial and GPU-rendered imagery to the MLSEC-based crease extraction algorithm to extract terrain valleys, which are presented in Fig. 5.2a and Fig. 5.2b, respectively. Differences between these two valley images can be explained by the rendering noise.



(a) Aerial terrain image (b) GPU render of GRRR data

Figure 5.1: Image data of Hualapai Peak at 10m resolution



(a) Terrain valleys in aerial imagery (b) Terrain valley in GPU render

Figure 5.2: Terrain valleys of Hualapai Peak at 10m resolution

5.1.1.1 Terrain Dataset

We divide both aerial and GPU-rendered terrain imagery, as shown in Fig. 5.1a and Fig. 5.1b, into 3×3 non-overlapping blocks. Size of each image block is equal to 180 by

180 pixels. That means we have 9 images over the terrain area from monocular camera and GPU render, respectively. Each pair of images, from different sources but with the same row-major order index, cover the same terrain block. We input 18 images to our terrain matching algorithm to measure similarity between 9 terrain blocks. Concretely, every time we first choose a pair of terrain images, one from camera and the other from GPU render, and input them to the minutiae extraction algorithm to obtain valley endings and bifurcations. We then input these two minutiae sets to our matching algorithm to measure similarity between two input terrain blocks. Overall we have a total of $\binom{9}{2} + 9 = 45$ match samples, among which 9 samples are PVA match and the other 36 are Non-PVA matches.

For each terrain block, number of minutiae in aerial and GPU-rendered images might be different due to noises in the rendering process. We therefore define minutiae number in one terrain block as the average of minutiae numbers in aerial and GPU-rendered images. Minutiae numbers in nine terrain blocks are depicted in Fig. 5.3. In this plot, x-axis and y-axis represent index of terrain blocks and values of minutiae number in nine terrain blocks, respectively. We make the observation that minutiae numbers in nine blocks range from 55 to 100, and average minutiae number over nine terrain blocks is equal to 80.

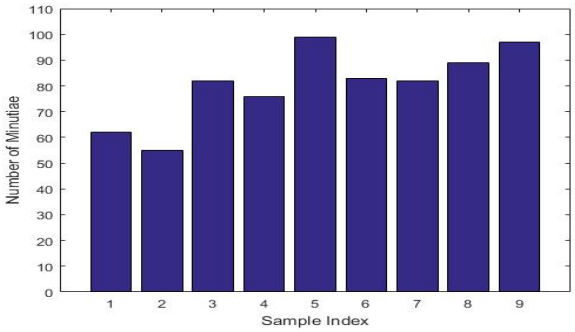


Figure 5.3: Numbers of minutiae in nine terrain blocks.

5.1.1.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching experiment over Hualapai Peak in Fig. 5.4, where x-axis represents row-major index of input image from camera, y-axis index of input image from GPU render, and z-axis similarity score between two input terrain blocks.

In this plot, the diagonal represent PVA matches, where higher bars indicate high certainty. Off-diagonal are Non-PVA matches, where a field-of-view matches with the terrain block that the aircraft is not immediately flying over.

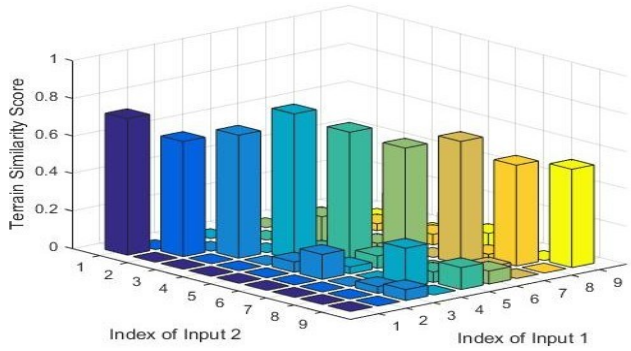


Figure 5.4: Similarity scores between nine input terrain blocks.

We employ nine samples on diagonal of Fig. 5.4 to build frequency histogram of similarity scores of PVA matches. The histogram is presented in Fig. 5.5. In this plot, values in x-axis are similarity scores divided into equally spaced bins, with width of each bin equal to 0.05. Values in y-axis represent numbers of PVA matches in these bins. We make the observation from Fig. 5.5 that similarity scores of nine PVA matches rangs from 0.50 to 0.80 in this experiment. Difference in similarity scores among nine terrain blocks can be explained by variance of feature strength in these blocks. Specifically, the stronger drainage pattern features in one terrain block, the larger similarity score we obtain from the PVA match.

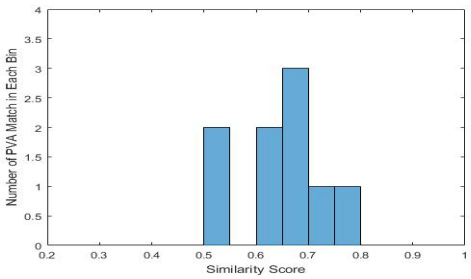


Figure 5.5: Frequency histogram of PVA similarity score

For Non-PVA match, we employ 36 off-diagonal similarity scores in Fig. 5.4 to build the frequency histogram. The histogram in presented in Fig. 5.6. In this plot, values in x-axis are similarity scores divided into equally spaced bins, with width of each bin equal to 0.05. Values in y-axis represent numbers of Non-PVA matches in these bins. We make the observation from

the plot that similarity scores of 36 Non-PVA matches are all smaller than 0.25. Especially, nearly 80 percentage of similarity scores are smaller than 0.05. This can be explained by large variation in numbers of minutiae over nine terrain blocks, as described in Fig. 5.3.

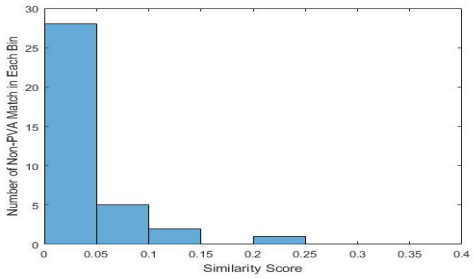


Figure 5.6: Frequency histogram of Non-PVA similarity score

In Chapter 4, we discuss value selection of matching threshold $TSimi$, and experimentally set $TSimi$ as 0.35. By setting $TSimi$ as 0.35, we conclude that our approach successfully recognizes all nine terrain blocks while producing zero false positives. That means when the terrain is identidied as the active flight region, all nine terrain blcoks are qualified to be selected as landmarks. Overall match results of our approach are summarized in Table. 5.1.

Table 5.1: Match results over Hualapai Peak at 10m resolution

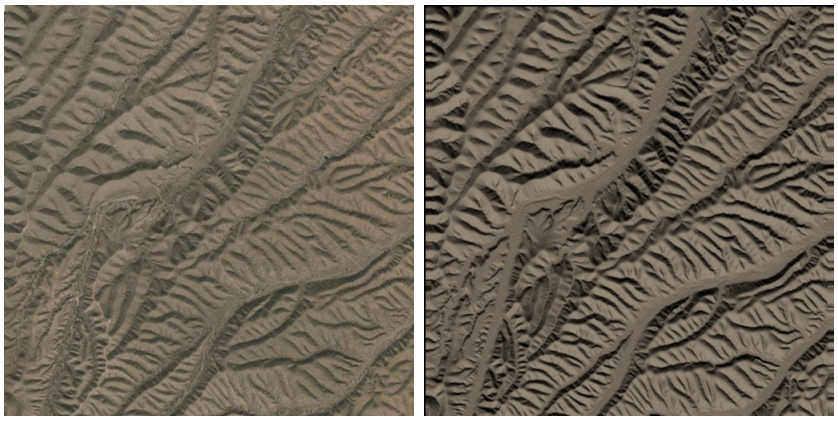
Detection Rate	**Miss Rate**	**False Positive Rate**
$9/9 = 100\%$	$0/9 = 0\%$	$0/36 = 0\%$

5.1.2 Kings Peak

The terrain area we consider for this experiment is presented in Fig. 5.7. Fig. 5.7a and Fig. 5.7b show aerial and GPU-rendered image of the terrain, both of which are of size 512 by 512 pixels. To facilitate observation of crease distribution, we present terrain valleys extracted from aerial and GPU-rendered imagery in Fig. 5.8a and Fig. 5.8b, respectively.

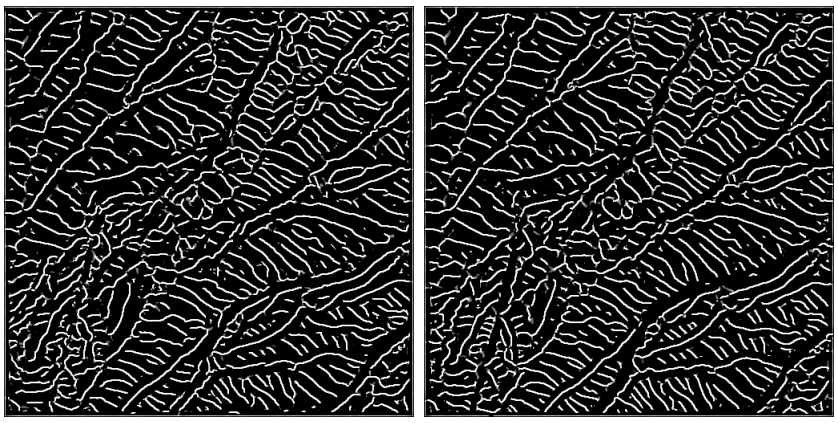
5.1.2.1 Terrain Dataset

We repeat the input generation process as described in Section 5.1.1.1 to generate 9 pairs of images as inputs of the experiment. Size of images are all equal to 180 by 180 pixels. Each pair



(a) Aerial terrain image (b) GPU render of GRRR data

Figure 5.7: Image data of Kings Peak at 10m resolution



(a) Terrain valleys in aerial imagery (b) Terrain valley in GPU render

Figure 5.8: Terrain valleys of Kings Peak at 10m resolution

of images from different sources but with the same index cover the same terrain block. Minutiae numbers in nine terrain blocks are shown in Fig. 5.9. We observe that minutiae numbers range from 75 to 115. Average minutiae number over nine blocks is equal to 93.

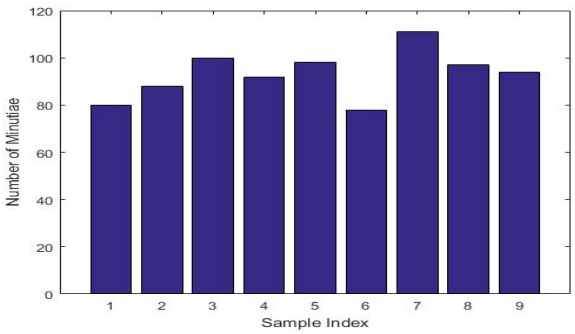


Figure 5.9: Numbers of minutiae in nine terrain blocks.

5.1.2.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching algorithm in Fig. 5.10, where x represents row-major index of input image from camera, y index of input image from GPU render, and z similarity score between two input terrain blocks.

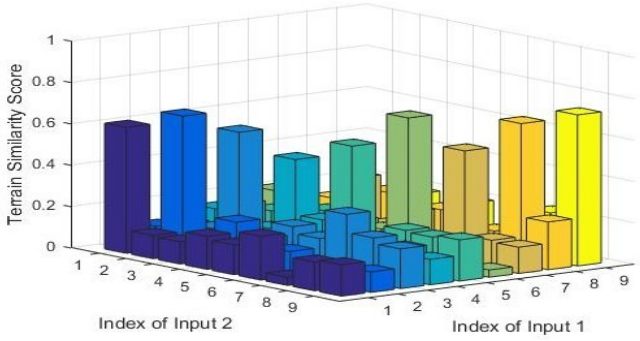


Figure 5.10: Similarity scores between nine terrain blocks.

We employ nine samples on diagonal of Fig. 5.10 to build frequency histogram of similarity scores of PVA matches. The histogram is presented in Fig. 5.11. We make the observation from Fig. 5.11 that similarity scores of nine PVA matches range from 0.45 to 0.75 in this experiment. Concretely, similarity scores of all other eight PVA matches are larger than 0.50 excepet for the PVA match corresponding to block 4. This can be explained by the weak drainage pattern features at right-bottom part of the terrain block.

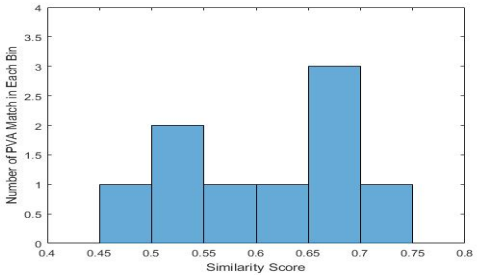


Figure 5.11: Frequency histogram of PVA similarity score

For Non-PVA match, we employ 36 off-diagonal similarity scores in Fig. 5.10 to build the frequency histogram. The histogram is presented in Fig. 5.12. We make the observation that similarity scores from 36 Non-PVA matches range from 0.0 to 0.35. Especially, 97.22 percentage of similarity scores is smaller than 0.25.

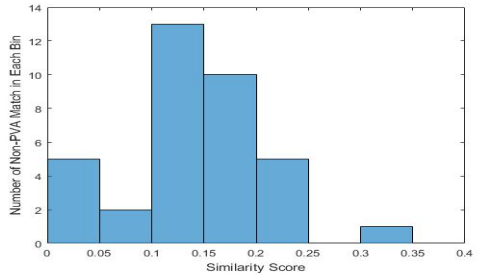


Figure 5.12: Frequency histogram of Non-PVA similarity score

Based on Fig. 5.11 and Fig. 5.12, we conclude that our approach successfully recognizes all nine terrain blocks with zero false positives by setting matching threshold $TSimi$ as 0.35. That means when the terrain is identidied as the active flight region, all nine terrain blcoks are qualified to be selected as landmarks. Match results of our approach over Kings Peak are summarized in Table. 5.2.

Table 5.2: Match results over Kings Peak at 10m resolution

Detection Rate	**Miss Rate**	**False Positive Rate**
$9/9 = 100\%$	$0/9 = 0\%$	$0/36 = 0\%$

5.2 5m Resolution

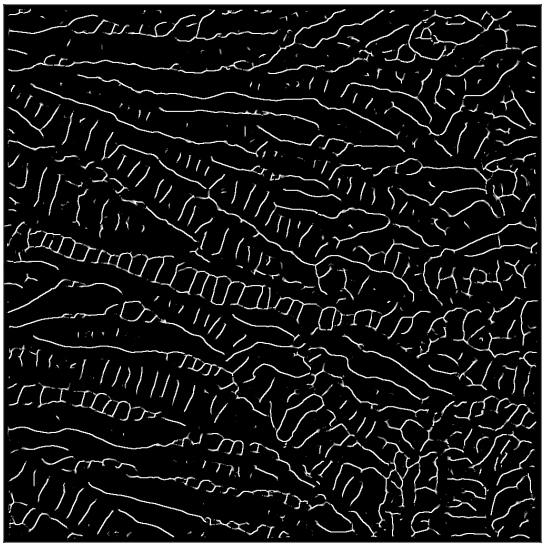
We conduct two experiments to measure similarity among terrain blocks over the same two terrain areas as shown in Fig. 5.1 and Fig. 5.7. Difference between these two experiments and those two in Section 5.1 is that these two experiments are conducted at the flight altitude such that each pixel on 2D image plane corresponds to one terrain cell of size 5m × 5m.

5.2.1 Hualapai Peak

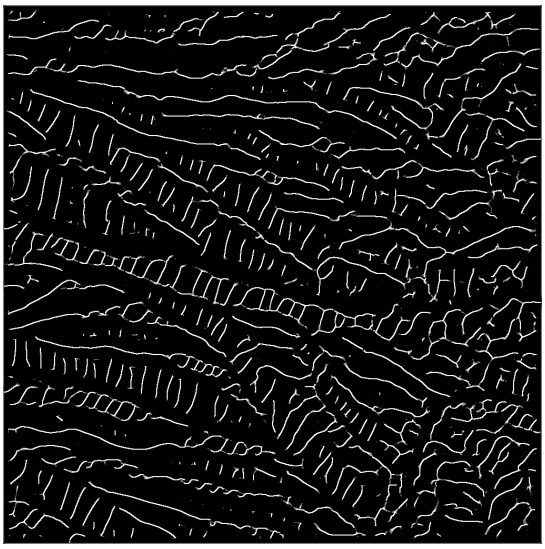
When UASs operate at altitude such that each pixel in aerial imagery corresponds to one terrain cell of size 5m × 5m, terrain area as shown in Fig. 5.1a will be mapped to an aerial image of size 1024 by 1024 pixels. To generate life-like appearance of the same terrain area, we first interpolate the 10m DEM provided by USGS to 5m resolution using IDW interpolation method. We then render them to realistic terrain model via Per-Pixel Displacement Mapping. To facilitate observation of crease distribution in this area, we input both aerial and GPU-rendered imagery to the MLSEC-based crease extraction process to extract terrain valleys, which are presented in Fig. 5.13a and Fig. 5.13b, respectively.

5.2.1.1 Terrain Dataset

We divide both aerial and GPU-rendered imagery of the terrain area into 6 × 6 non-overlapping blocks, with size of each image block equal to 180 by 180 pixels. That means we have 36 images over the terrain area from on-board camera and GPU render, respectively. Each pair of images, from different sources but with the same row-major order index, cover the same terrain block. It is worth mentioning that although image size of each terrain block is the same as that at 10m resolution, size of a terrain block is actually one fourth of that at 10m resolution due to higher image resolution. We input 72 images to our terrain matching algorithm to measure similarity between 36 terrain blocks. Concretely, every time we first choose a pair of terrain images, one from camera and the other from GPU render, and input them to the minutiae extraction process to obtain valley endings and bifurcations. We then input these two minutiae sets to our matching algorithm to measure similarity between two input terrain



(a) Terrain valleys in aerial imagery

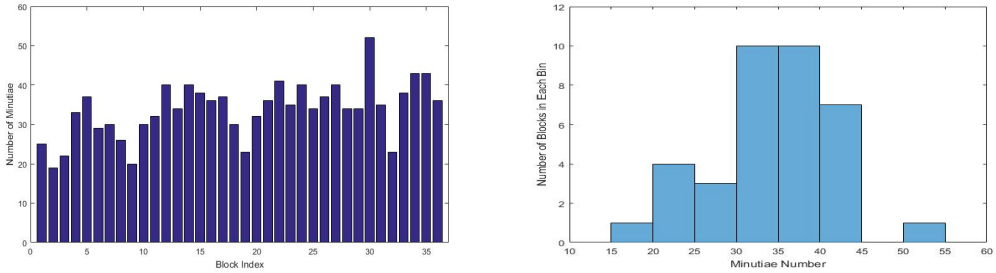


(b) Terrain valleys in GPU render

Figure 5.13: Terrain valleys of Hualapai Peak at 5m resolution. Sizes of both valley images are equal to 1024 by 1024 pixels.

blocks. Overall we have a total of $\binom{36}{2}$ +36 = 666 match samples, among which 36 samples are PVA match and the other 630 are Non-PVA matches.

Minutiae numbers in 36 blocks are depicted in Fig. 5.14a. Besides, we build a frequency histogram as given in Fig. 5.14b to show distribution of minutiae numbers. In this plot, x-axis represents number of minutiae divided into equally spaced bins with width of each bin equal to 5, and y represents number of terrain blocks in each bin. We observe that minutiae numbers range from 19 to 52. Among 36 samples, minutiae numbers in 27 blocks are between 30 and 45. Average minutiae number over 36 terrain blocks is equal to 34. In addition, by comparing Fig. 5.14 with Fig. 5.3, we observe that average minutiae number in a fixed-size terrain image at 5m resolution is smaller than that at 10m resolution due to smaller size of terrain blocks.



(a) Minutiae numbers in 36 terrain blocks (b) Frequency histogram of minutiae numbers

Figure 5.14: Minutiae numbers in 36 terrain blocks.

5.2.1.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching approach in Fig. 5.15, where x-axis represents index of input crease block from camera, y-axis index of input crease block from GPU render, and z similarity score between two input terrain blocks.

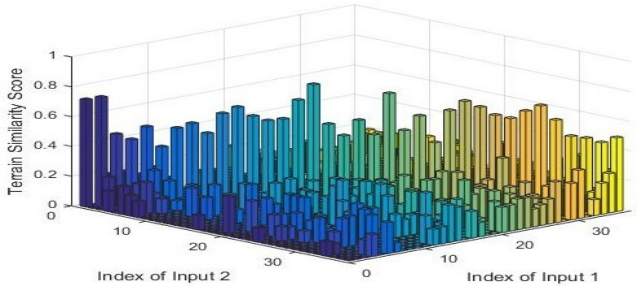


Figure 5.15: Similarity scores between 36 input terrain blocks.

We employ 36 samples on diagonal of Fig. 5.15 to build frequency histogram of similarity scores of PVA matches. The histogram is presented in Fig. 5.16. To facilitate observation, we also summarize number of PVA matches in each bin in Table 5.3. Based on Table 5.3, we make the observation that similarity scores ranges from 0.40 to 0.85 in this experiment. By comparing Fig. 5.16 with Fig. 5.5, we notice that range of similarity scores is slightly larger than that of 10m resolution. This can be explained by the fact that difference in average strength of drainage patterns among these terrain blcoks gets larger as size of terrain blocks decreases.

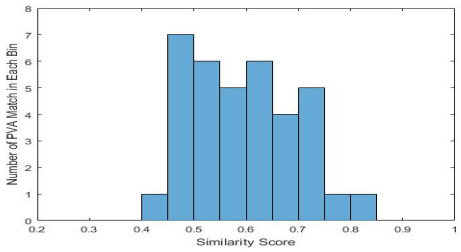


Figure 5.16: Frequency histogram of PVA similarity score

Table 5.3: Distribution of similarity scores from PVA matches

Bin Locations	$[0.40, 0.45)$	$[0.45, 0.50)$	$[0.50, 0.55)$	$[0.55, 0.60)$	$[0.60, 0.65)$	$[0.65, 0.70)$	$[0.70, 0.75)$	$[0.75, 0.80)$	$[0.80, 0.85)$
Counts	1	7	6	5	6	4	5	1	1
Frequency	0.0278	0.1944	0.1667	0.1389	0.1667	0.1111	0.1389	0.0278	0.00278

For Non-PVA match, we utilize 630 off-diagonal similarity scores in Fig. 5.15 to build the frequency histogram. The histogram is presented in Fig. 5.17. To facilitate observation, we also summarize counts of Non-PVA matches in each bin in Table 5.4. Based on Table 5.4, we make the observation that similarity scores of 629 Non-PVA matches are smaller than 0.35. There is only one PVA match whose similarity score is between 0.35 and 0.40.

Table 5.4: Distribution of similarity scores from Non-PVA matches

Bin Locations	$[0.00, 0.05)$	$[0.05, 0.10)$	$[0.10, 0.15)$	$[0.15, 0.20)$	$[0.20, 0.25)$	$[0.25, 0.30)$	$[0.30, 0.35)$	$[0.35, 0.40)$
Counts	422	36	62	48	41	17	3	1
Frequency	0.6698	0.0571	0.0984	0.0762	0.0651	0.0270	0.0048	0.0016

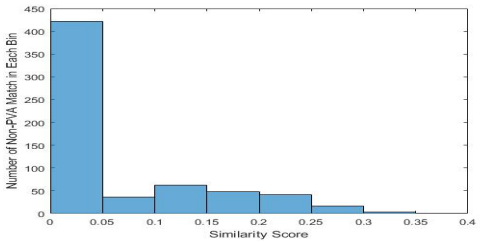


Figure 5.17: Frequency histogram of Non-PVA similariy score

By setting $TSimi$ as 0.35, we conclude that our approach successfully recognizes all 36 terrain blocks while producing only 1 false positives. Match results are summarized in Table. 5.5.

Table 5.5: Match results over Hualapai Peak at 5m resolution

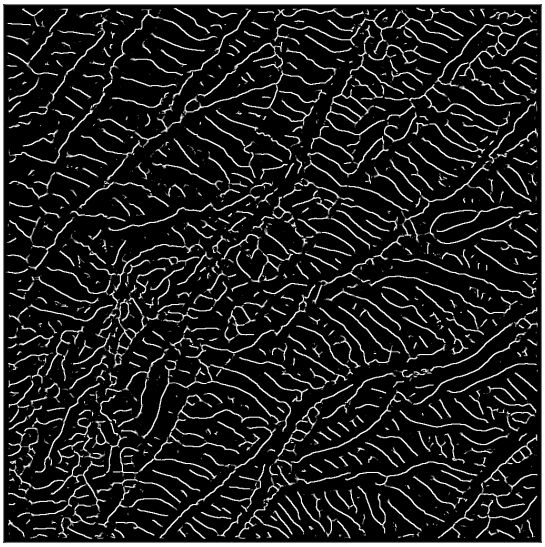
Detection Rate	**Miss Rate**	**False Positive Rate**
$36/36 = 100\%$	$0/36 = 0$	$1/630 = 0.16\%$

5.2.2 Kings Peak

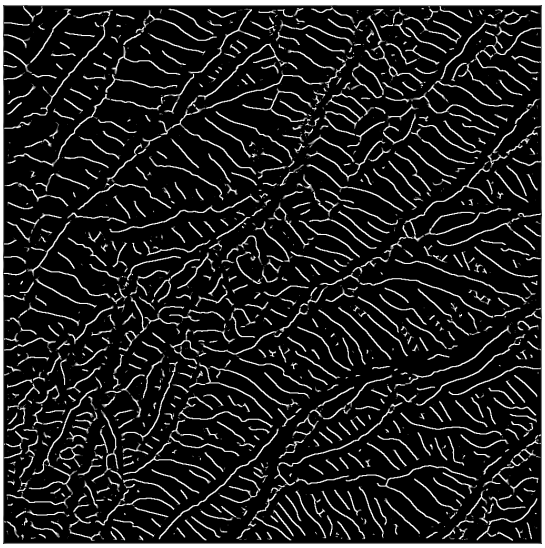
When UASs operate at altitude such that each pixel in aerial imagery corresponds to one terrain cell of size 5m $\times$ 5m, terrain area as shown in Fig. 5.7a will be mapped to an image of size 1024 by 1024 pixels. We employ the same technique as described in Section 5.2.1 to generate life-like appearance of the area using GRRR data. Terrain valleys in aerial and GPU-rendered images are presented in Fig. 5.18a and Fig. 5.18b, respectively.

5.2.2.1 Terrain Dataset

We repeat the input generation process as described in Section 5.2.1.1 to generate 36 pairs of images as inputs of the experiment. Size of images are all equal to 180 by 180 pixels. Each pair of images from different sources but with the same row-major index cover the same terrain block. Minutiae numbers in 36 terrain blocks are depicted in Fig. 5.19a. We observe that minutiae numbers ranges from 29 to 64, and average minutiae number over 36 terrain blocks is equal to 46.

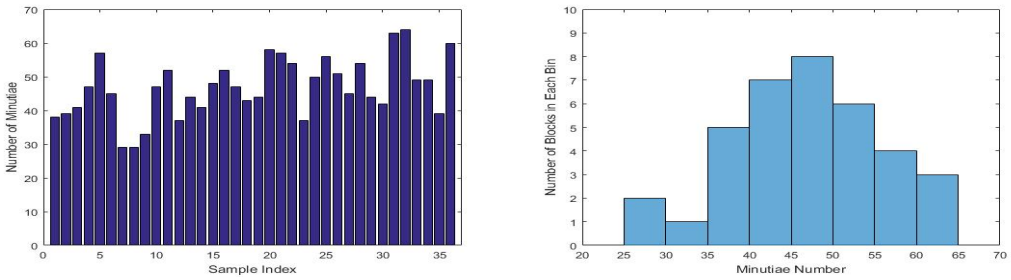


(a) Terrain valleys in aerial imagery



(b) Terrain valley in GPU render

Figure 5.18: Terrain valleys of Kings Peak at 5m resolution. Sizes of both crease images are equal to 1024 by 1024 pixels.



(a) Minutiae numbers in 36 terrain blocks (b) Frequency histogram of minutiae numbers

Figure 5.19: Minutiae numbers in 36 terrain blocks.

5.2.2.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching approach in Fig. 5.20, where x represents index of input crease block from camera, y index of input crease block from GPU render, and z similarity score between two input terrain blocks.

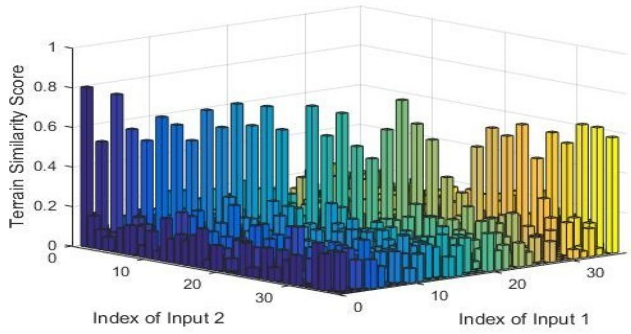


Figure 5.20: Similarity scores between 36 terrain blocks.

We employ 36 samples on diagonal of Fig. 5.20 to build frequency histogram of similarity scores of PVA matches, and show the histogram in Fig. 5.21. Besides, we summarize counts of PVA matches in each bin in Table 5.6. Based on Table 5.6, we make the observation that simility scores of PVA matches ranges from 0.20 to 0.85. In detail, similarity scores of 34 PVA matches are larger than 0.35, and similarity scores of the other two PVA matches are between 0.20 and 0.30. Rectangles in Fig. 5.22 represent the two terrain blocks associated with the PVA matches whose similarity score are smaller than 0.30. By checking these two blocks, we learn that the small similarity scores can be explained by weak terrain valleys in these two blocks.

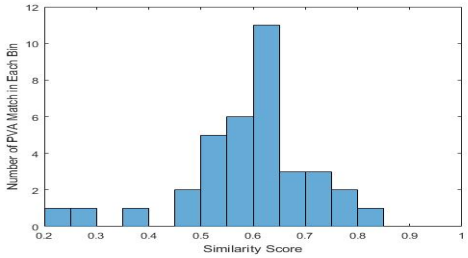


Figure 5.21: Frequency histogram of PVA similarity score

Table 5.6: Distribution of similarity scores from PVA matches

Bin Locations	$[0.20, 0.30)$	$[0.35, 0.40)$	$[0.45, 0.50)$	$[0.50, 0.55)$	$[0.55, 0.60)$	$[0.60, 0.65)$	$[0.65, 0.70)$	$[0.70, 0.75)$	$[0.75, 0.80)$	$[0.80, 0.85)$
Counts	2	1	2	5	6	11	3	3	2	1
Frequency	0.0556	0.0278	0.0556	0.1389	0.1667	0.3056	0.0833	0.0833	0.0556	0.0278

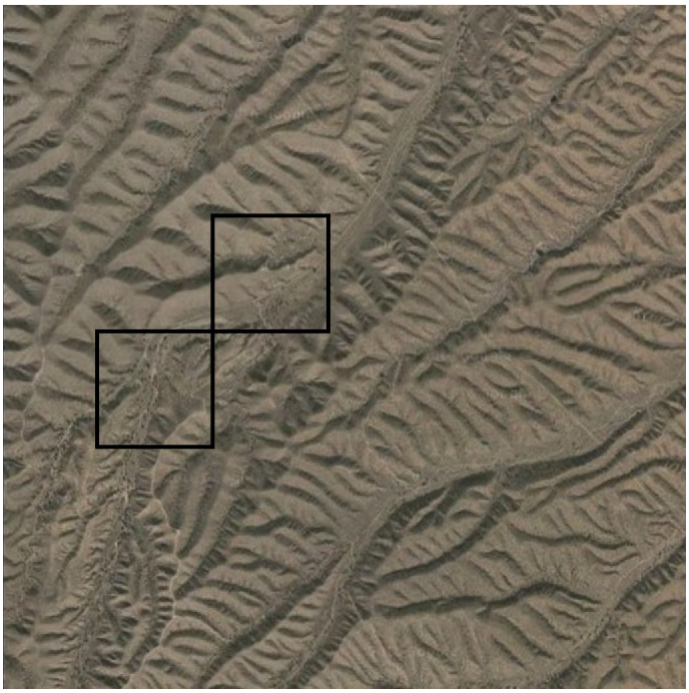


Figure 5.22: Two terrain blocks missed by our terrain matching algorithm.

For Non-PVA match, we utilize 630 off-diagonal similarity scores in Fig. 5.20 as samples to build a frequency histogram. The histogram is presented in Fig. 5.23. To facilitate observation, we also summarize counts of Non-PVA matches in each bin in Table 5.7. Based on Table 5.7, we make the observation that similarity scores ranges from 0.0 to 0.35. It's worth mentioning that a large number (i.e., 49.5%) of similarity scores is smaller 0.05. This can be explained by the large difference of minutiae number among these terrain blocks as described in Fig. 5.19.

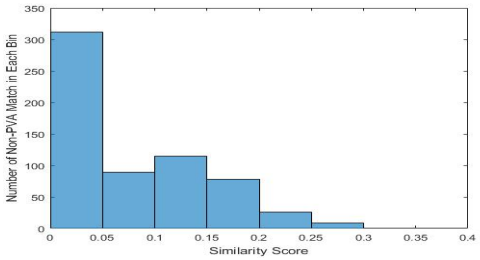


Figure 5.23: Frequency histogram of Non-PVA similarity score

Table 5.7: Distribution of similarity scores from Non-PVA matches

Bin Locations	$[0.00, 0.05)$	$[0.05, 0.10)$	$[0.10, 0.15)$	$[0.15, 0.20)$	$[0.20, 0.25)$	$[0.25, 0.30)$	$[0.30, 0.35)$
Counts	312	89	115	78	26	9	1
Frequency	0.4952	0.1413	0.1825	0.1238	0.0413	0.0143	0.0016

By setting matching threshold $TSimi$ as 0.35, we conclude that our approach misses two terrain blocks while producing zero false positives. Match results of our approach are summarized in Table. 5.8. The misses are accpetable due to weak drainage patterns in these blocks.

Table 5.8: Match results over Kings Peak at 5m resolution

Detection Rate	**Miss Rate**	**False Positive Rate**
$34/36 = 94.44\%$	$2/36 = 5.56\%$	$0/630 = 0\%$

5.3 2.5m Resolution

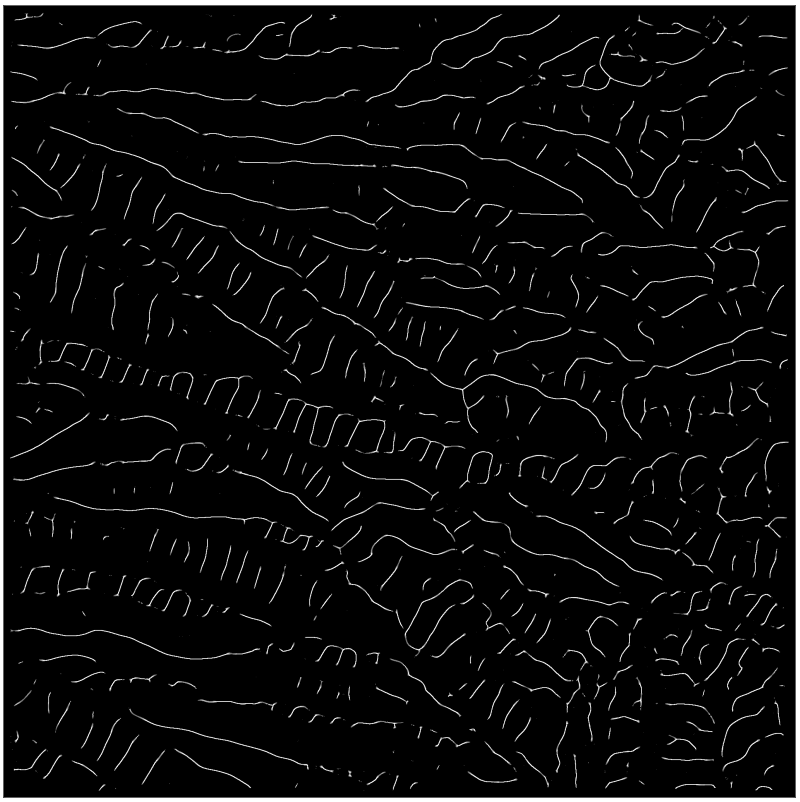
Two experiments are designed to measure similarity among terrain blocks over the two terrain areas as shown in Fig. 5.1 and Fig. 5.7. Both experiments are conducted at the flight altitude such that each pixel corresponds to one terrain cell of size 2.5m × 2.5m.

5.3.1 Hualapai Peak

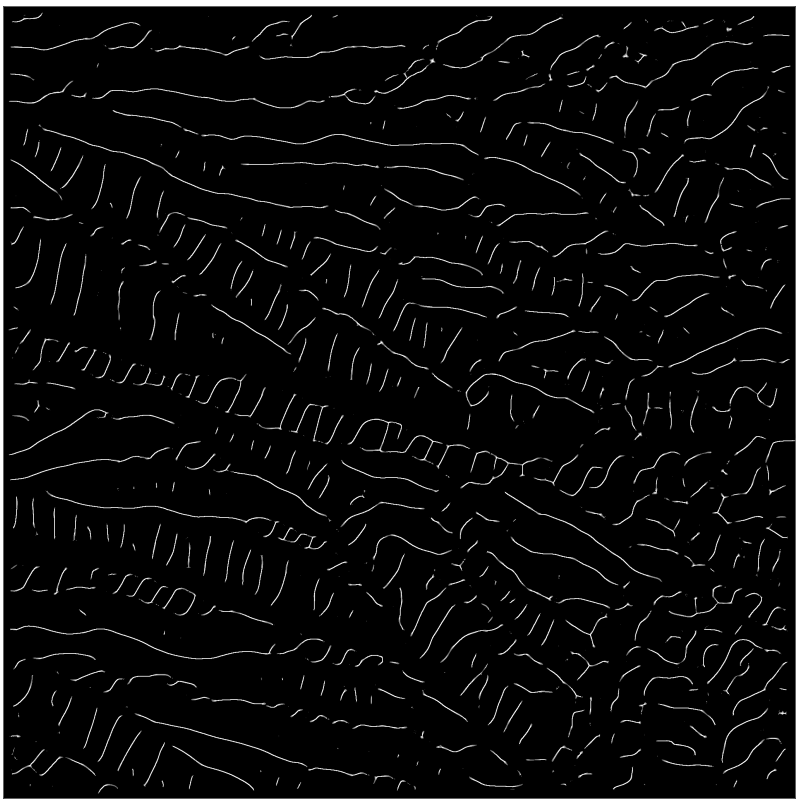
When UASs operate at altitude such that each pixel in aerial imagery corresponds to one terrain cell of size 2.5m × 2.5m, terrain area as shown in Fig. 5.1a will be mapped to an aerial image of size 2048 by 2048 pixels. To generate life-like appearance of the terrain area, we first interpolate the 10m DEM to 2.5m resolution, and then render them to realistic terrain model via Per-Pixel Displacement Mapping. To facilitate observation of crease distribution in this area, we input both aerial and GPU-rendered imagery to the MLSEC-based crease extraction process to extract terrain valleys, which are presented in Fig. 5.24a and Fig. 5.24b, respectively.

5.3.1.1 Terrain Dataset

We divide both aerial and GPU-rendered imagery of the terrain into 12×12 non-overlapping blocks, with size of each block equal to 180 by 180 pixels. That means we have 144 images over the terrain area from camera and GPU render, respectively. Each pair of images, from different sources but with the same row-major order index, cover the same terrain block. It is worth mentioning that although image size of each terrain block is the same as that at both 5m and 10m resolution, size of a terrain block is actually one fourth of that at 5m resolution and one sixteenth of that at 10m resolution due to high image resolution. We input 2×144 images to our terrain matching algorithm to measure similarity between 144 terrain blocks. Concretely, every time we first choose a pair of terrain images, one from camera and the other from GPU render, and input them to the minutiae extraction process to obtain valley endings and bifurcations. We then input these two minutiae sets to our matching algorithm to measure similarity between two inputs. Overall we have a total of $\binom{144}{2} + 144 = 10440$ match samples, among which 144 samples are PVA match and the other 10296 are Non-PVA matches.



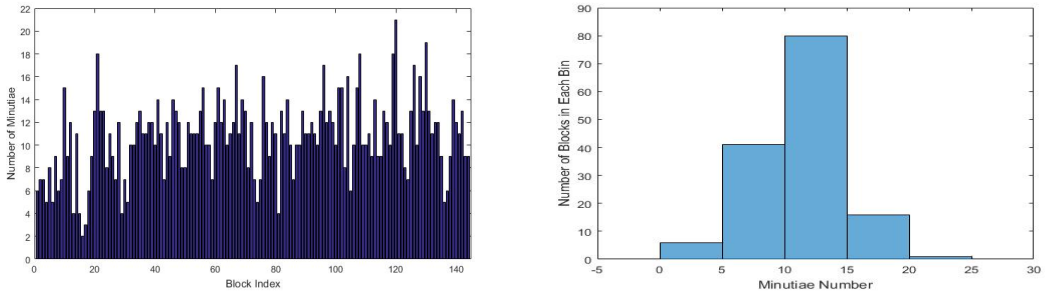
(a) Terrain valleys in aerial imagery



(b) Terrain valley in GPU render

Figure 5.24: This figure shows terrain valleys of Hualapai Peak at 2.5m resolution. Sizes of both valley images are equal to 2048 by 2048 pixels.

Minutiae numbers in 144 terrain blocks over the terrain are depicted in Fig. 5.25. Besides, we build the frequency histogram as given in Fig. 5.25b to show distribution of minutiae numbers over 144 blocks. We observe from these two plots that minutiae numbers range from 2 to 21. In detail, 84 percentage of minutiae numbers are between 5 and 15. Average minutiae number over 144 terrain blocks is equal to 11. By comparing Fig. 5.25 with Fig. 5.3 and Fig. 5.14, we observe that average minutiae number in a fixed-size terrain image at 2.5m resolution is smaller than that at both 5m and 10m resolution due to smaller size of terrain blocks.



(a) Minutiae numbers in 144 terrain blocks (b) Frequency histogram of minutiae numbers

Figure 5.25: Minutiae numbers in 144 terrain blocks.

5.3.1.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching algorithm in Fig. 5.26, where x-axis represents row-major index of input terrain image from camera, y-axis index of input image from GPU render, and z similarity score between two input terrain blocks.

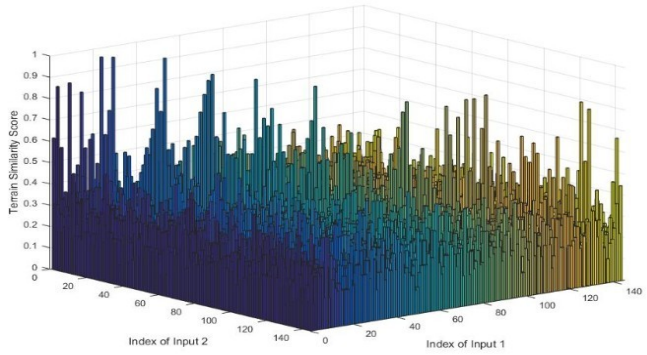


Figure 5.26: Similarity scores between 144 input terrain blocks.

We employ 144 samples on diagonal of Fig. 5.26 to build frequency histogram of similarity scores of PVA matches. The histogram is presented in Fig. 5.27. To facilitate observation, we also summarize counts of PVA matches in each bin in Table 5.9. Based on Table 5.9, we make the observation that similarity scores of 144 PVA matches range from 0 to 1 in this experiment. By comparing Fig. 5.27 with Fig. 5.16 and Fig. 5.5, we notice obvious increase in range of similarity scores from PVA matches. This can be explained as follows. Number of drainage patterns descreas with size of terrain blocks. In this scenario, size of terrain blocks descreses to an extent such that there exist some terrain blocks in which drainage patterns are all weak, or all strong. Similarity scores from these terrain blocks increase range of similarity scores. Besides, we observe that among 144 samples, 24 similarity scores are smaller than 0.35.

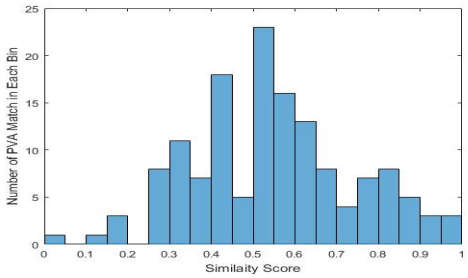


Figure 5.27: Frequency histogram of PVA similarity score

Table 5.9: Distribution of similarity scores from PVA matches

Bin Locations	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**
$[0.00, 0.05)$	1	0.0069	$[0.40, 0.45)$	18	0.1250	$[0.70, 0.75)$	4	0.0278
$[0.10, 0.15)$	1	0.0069	$[0.45, 0.50)$	5	0.0347	$[0.75, 0.80)$	7	0.0486
$[0.15, 0.20)$	3	0.0208	$[0.50, 0.55)$	23	0.1597	$[0.80, 0.85)$	8	0.0556
$[0.25, 0.30)$	8	0.0556	$[0.55, 0.60)$	16	0.1111	$[0.85, 0.90)$	5	0.0347
$[0.30, 0.35)$	11	0.0764	$[0.60, 0.65)$	13	0.0903	$[0.90, 0.95)$	3	0.0208
$[0.35, 0.40)$	7	0.0486	$[0.65, 0.70)$	8	0.0556	$[0.95, 0.10]$	3	0.0208

For Non-PVA match, we utilize 10296 off-diagonal similarity scores in Fig. 5.26 to build the frequency histogram. The histogram is presented in Fig. 5.28. To facilitate observation, we

also summarize counts of Non-PVA matches in each bin in Table 5.10. Based on Table 5.10, we observe that similarity scores of Non-PVA matches range from 0 to 0.65. In addition, among 10296 Non-PVA matches, 4 percentage of similarity scores is larger than 0.35.

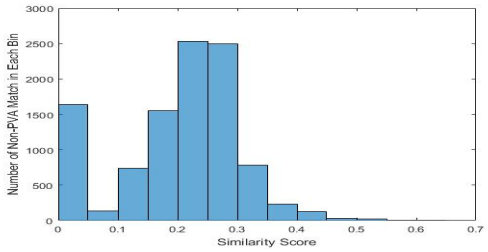


Figure 5.28: Frequency histogram of Non-PVA similarity score

Table 5.10: Distribution of similarity scores from Non-PVA matches

Bin Locations	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**
$[0.00, 0.05)$	1637	0.1590	$[0.25, 0.30)$	2494	0.2422	$[0.50, 0.55)$	21	0.0020
$[0.05, 0.10)$	136	0.0132	$[0.30, 0.35)$	784	0.0761	$[0.55, 0.60)$	1	0.0001
$[0.10, 0.15)$	741	0.0720	$[0.35, 0.40)$	232	0.0225	$[0.60, 0.65)$	3	0.0003
$[0.15, 0.20)$	1556	0.1511	$[0.40, 0.45)$	128	0.0124			
$[0.20, 0.25)$	2536	0.2463	$[0.45, 0.50)$	27	0.0026			

5.3.1.3 Performance Analysis

Based on Table 5.9 and 5.10, we conclude that our approach successfully recognizes 120 terrain blocks while producing 412 false positives by setting setting $TSimi$ as 0.35. Match results are summarized in Table. 5.11.

Table 5.11: Match results over Hualapai Peak at 2.5m Resolution

Detection Rate	**Miss Rate**	**False Positive Rate**
$120/144 = 83.33\%$	$24/144 = 16.67\%$	$412/10296 = 4.00\%$

We present 24 terrain misses in Fig. 5.29, where each black rectangle represents one missed terrain block. By checking these blocks, we make the observation that in these terrain blocks drainage patterns are weak and numbers of drainage patterns are small. It is worth mentioning that noise will be introduced during rendering process due to lighting condition. Weakness of drainage patterns combined with noise make our crease extraction process hard to accurately locate drainage patterns in GPU-rendered image of these blocks.

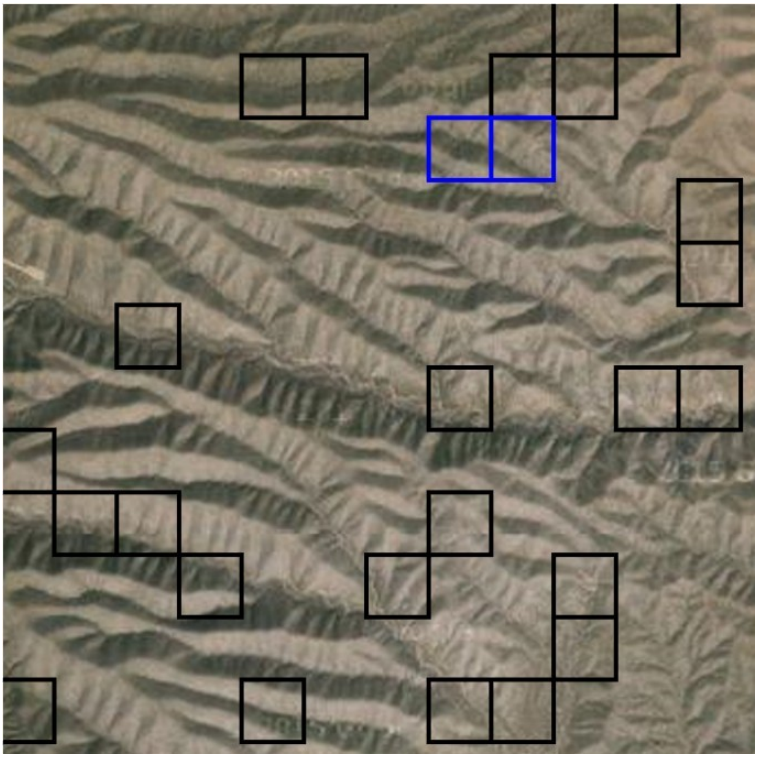
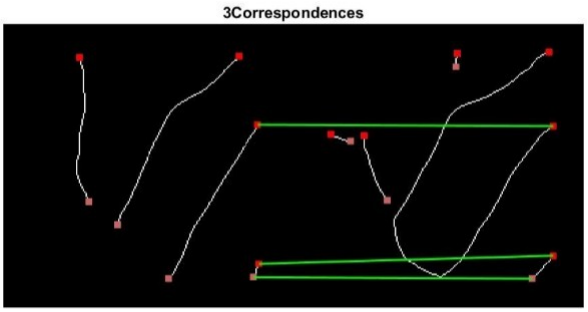


Figure 5.29: 25 terrain blocks missed by our terrain recognition approach

One example of terrain miss is shown in Fig. 5.30. Fig. 5.30a and Fig. 5.30c are image data from camera and GPU render, respectively. By observing the aerial image, we know that drainage patterns at bottom of the terrain block are weak. We apply MLSEC-based crease extraction on two images and present terrain valleys in Fig. 5.30b and Fig. 5.30d, respectively. Minutiae in these two crease blocks are shown in Fig. 5.30e. There are a total of 10 and 8 minutiae in these two minutiae sets, respectively. By comparing two minutiae sets, we make the observation that minutiae distribution at bottom are different due to rendering noise. We input these two minutiae sets to our matching algorithm, and obtain a total of 3 pairings. As

there are a total of 18 minutiae, similarity score between these two terrain blocks is calculated as 0.33.

(a) Aerial image (b) Aerial crease (c) GPU image (d) GPU crease



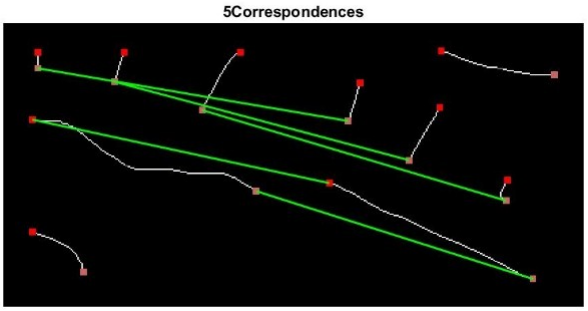
(e) Output of matching algorithm

Figure 5.30: An example of terrain miss

In this experiment, we obtain a total of 412 false positives among 10296 samples of Non-PVA match. These false positives can be explained by the following facts: (1) there are not sufficient minutiae and crease information in neighborhood for a given minutiae to identify itself as size of terrain blocks decreases; (2) correlation between contiguous blocks gets larger as size of terrain blocks decreases. One example of false positive is given in Fig. 5.31. Fig. 5.31a and Fig. 5.31c are image data of two different terrain blocks. Locations of two input terrain blocks are represented by blue rectangles in Fig. 5.29. We observe that the two blocks are contiguous to each other in row. We apply MLSEC-based terrain valley extraction on these two blocks, and present the extracted terrain valleys in Fig. 5.31b and Fig. 5.31d, respectively. Minutiae in these two valley images are shown in Fig. 5.31e. There are ten minutiae in each valley image. By comparing these two minutiae sets, we make the observation that spatial distribution of minutiae in two sets are similar to each other. We input two minutiae sets to

our minutiae-based matching algorithm, and obtain a total of five correspondences. Terrain similarity score between two terrain blocks is therefore calculated as 0.50.

(a) Block 32 (b) Crease in block 32 (c) Block 33 (d) Crease in block 33



(e) Output of matching algorithm

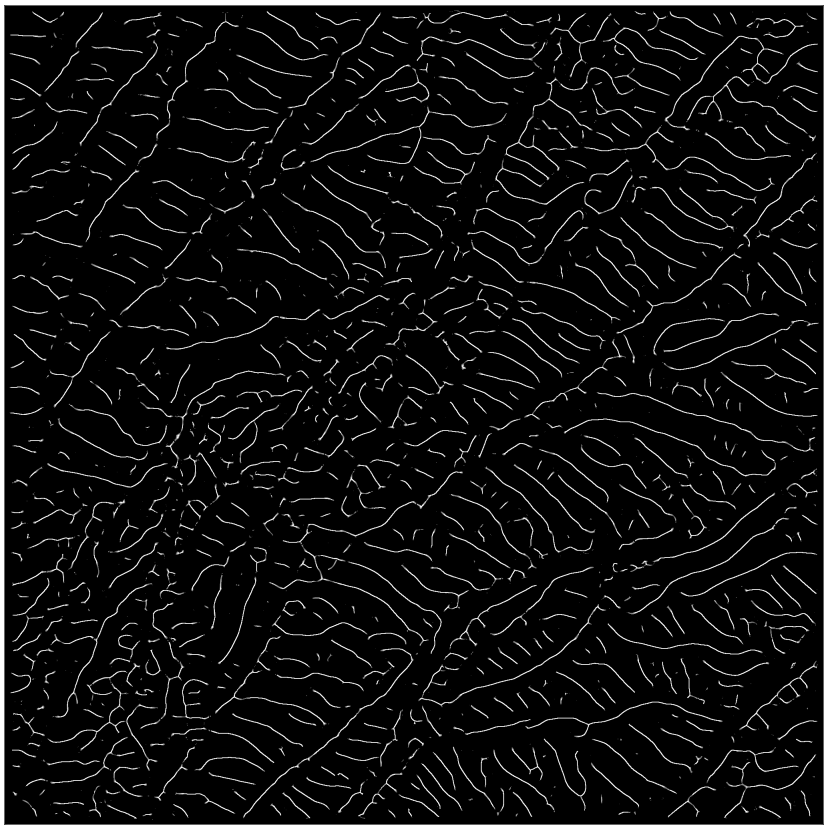
Figure 5.31: An example of terrain false positive

5.3.2 Kings Peak

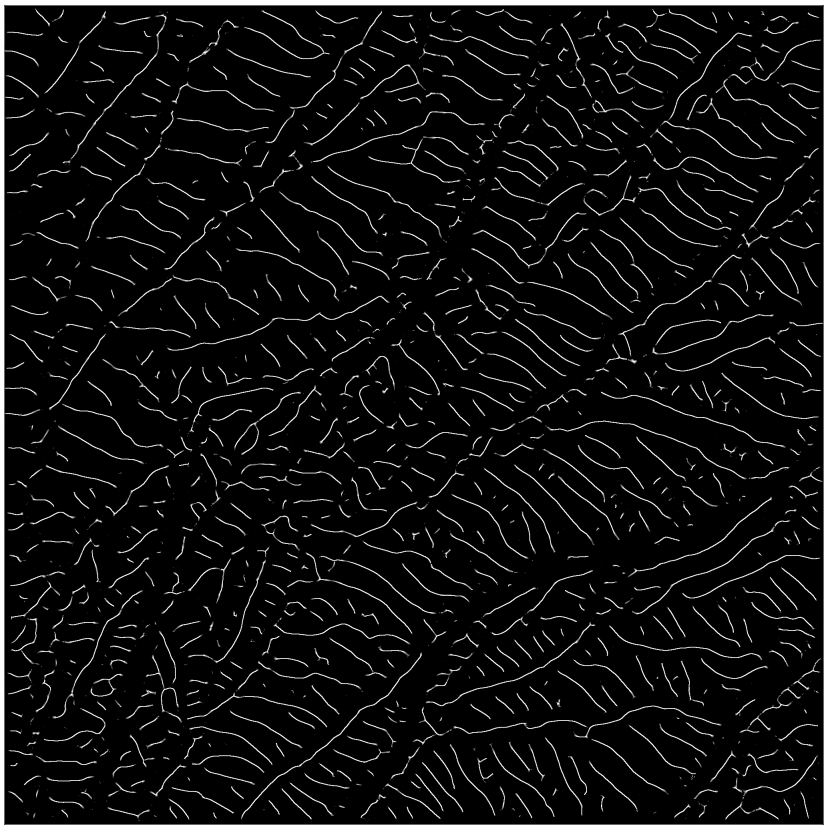
In this experiment, UAS operates over Kings Peak area at flight altitude such that each pixel in aerial imagery corresponds to one terrain cell of size 2.5m × 2.5m. Terrain area as shown in Fig. 5.7a will be mapped to an aerial image of size 2048 by 2048 pixels. We employ the same technique as described in Section 5.3.1 to generate life-like appearance of the same terrain using GRRR data. Terrain valleys from aerial imagery and GPU render are presented in Fig. 5.32a and Fig. 5.32b, respectively.

5.3.2.1 Terrain Dataset

We repeat the input generation process as described in Section 5.3.1.1 to generate 144 pairs of images as inputs of the experiment. Each pair of images, from different sources but with the



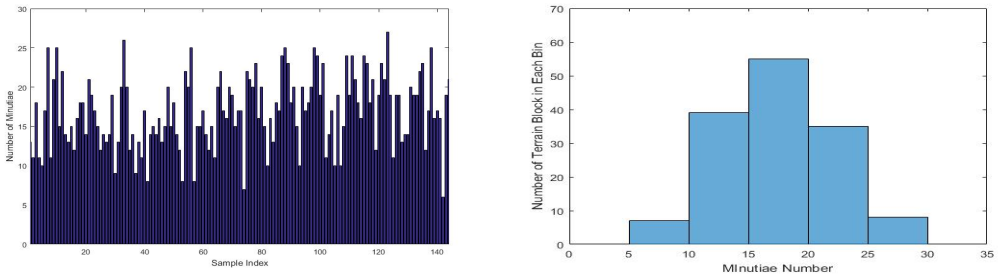
(a) Terrain valleys in aerial imagery



(b) Terrain valley in GPU render

Figure 5.32: Terrain valleys of Kings Peak at 2.5m resolution. Size of both valley image is equal to 2048 by 2048 pixels.

same row-major index, cover the same terrain block. Minutiae numbers in 144 terrain blocks are depicted in Fig. 5.33. We make the observation that minutiae numbers in 144 blocks range from 6 to 27. Average minutiae number over 144 terrain blocks is equal to 17. By comparing Fig. 5.33 with Fig. 5.9 and Fig. 5.19, we observe that average minutiae number in fixed-size terrain image of 2.5m resolution is smaller than that of both 10m and 5m resolution.



(a) Minutiae numbers in 144 terrain blocks (b) Frequency histogram of minutiae numbers

Figure 5.33: Minutiae numbers in 36 terrain blocks.

5.3.2.2 Minutiae-based Terrain Matching

We present outputs of our minutiae-based terrain matching approach in Fig. 5.34, where x-axis represents row-major index of input image from camera, y-axis index of input image from GPU render, and z similarity score between two input terrain blocks.

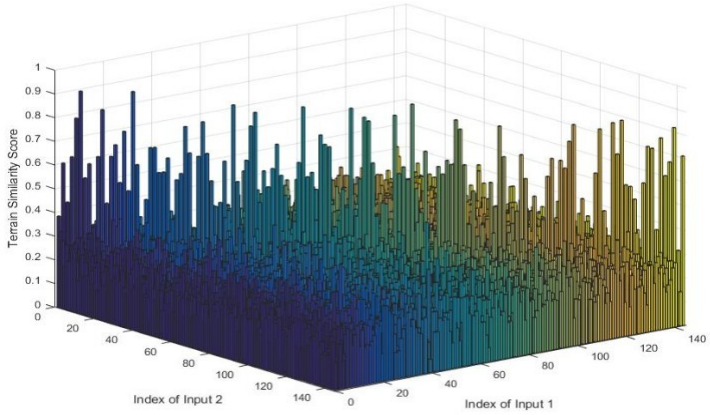


Figure 5.34: Similarity scores between 144 input terrain blocks.

We employ 144 samples on diagonal of Fig. 5.34 to build histogram of similarity scores of PVA match. The frequency histogram is presented in Fig. 5.35. To facilitate observation, we also summarize counts of PVA matches in each bin in Table 5.12. Based on Table 5.12,

we make the observation that similarity scores of 144 PVA matches range from 0.15 to 0.95. By comparing Fig. 5.35 with Fig. 5.11 and Fig. 5.21, we notice obvious increase in range of similarity scores. This can be explained as follows. Number of drainage patterns descreas with size of terrain blocks. In this scenario, size of terrain blocs descreses to an extent such that there exist some terrain blocks in which drainage patterns are all weak, or all strong. Similarity scores from these terrain blocks increase range of similarity scores. In addition, we observe that among 144 samples, similarity scores of 15 PVA mathes are smaller than 0.35.

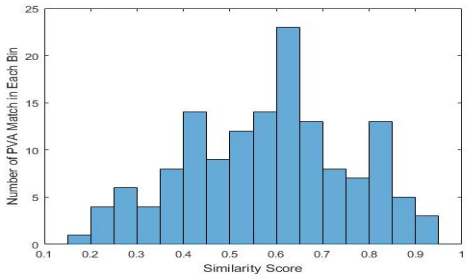


Figure 5.35: Frequency histogram of PVA similarity score

Table 5.12: Distribution of similarity scores from PVA matches

Bin Locations	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**
$[0.15, 0.20)$	1	0.0069	$[0.45, 0.50)$	9	0.0625	$[0.75, 0.80)$	7	0.0486
$[0.20, 0.25)$	4	0.0278	$[0.50, 0.55)$	12	0.0833	$[0.80, 0.85)$	13	0.0903
$[0.25, 0.30)$	6	0.0417	$[0.55, 0.60)$	14	0.0972	$[0.85, 0.90)$	5	0.0347
$[0.30, 0.35)$	4	0.0278	$[0.60, 0.65)$	23	0.1297	$[0.90, 0.95)$	3	0.0208
$[0.35, 0.40)$	8	0.0556	$[0.65, 0.70)$	13	0.0903			
$[0.40, 0.45)$	14	0.0972	$[0.70, 0.75)$	8	0.0556			

For Non-PVA match, we utilize 10296 off-diagonal similarity score in Fig. 5.34 to build a frequency histogram. The histogram is presented in Fig. 5.36. We also summarize counts of Non-PVA matches in each bin in Table 5.13. Based on Table 5.13, we make the observation that 98.65 percentage of similarity scores of Non-PVA matches is smaller than 0.35.

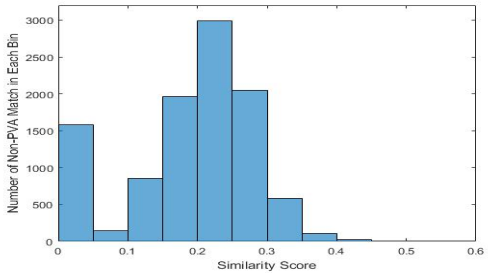


Figure 5.36: Frequency histogram of Non-PVA similarity score

Table 5.13: Distribution of similarity scores from Non-PVA matches

Bin Locations	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**	**Bin Locations**	**Counts**	**Frequency**
$[0.00, 0.05)$	1585	0.1539	$[0.20, 0.25)$	2991	0.2905	$[0.40, 0.45)$	27	0.0026
$[0.05, 0.10)$	141	0.0137	$[0.25, 0.30)$	2048	0.1989	$[0.45, 0.50)$	4	0.0004
$[0.10, 0.15)$	849	0.0825	$[0.30, 0.35)$	583	0.0566	$[0.50, 0.55)$	1	0.0001
$[0.15, 0.20)$	1961	0.1905	$[0.35, 0.40)$	106	0.0103			

5.3.2.3 Performance Analysis

Based on Table 5.12 and 5.13, we make the observation that by setting $TSimi$ as 0.35, our approach successfully recognizes 129 terrain blocks while producing only 138 false positives. Match results are summarized in Table 5.14.

Table 5.14: Match results over Kings Peak at 2.5m Resolution

Detection Rate	**Miss Rate**	**False Positive Rate**
$129/144 = 89.59\%$	$15/144 = 10.41\%$	$138/10296 = 1.35\%$

We present 15 terrain misses in Fig. 5.37, where each rectangle represents one miss. By checking these blocks, we observe that drainage patterns in these terrain blocks are weak. Weakness of drainage patterns combined with noise introduced by the rendering process make our crease extraction process hard to accuretely locate these creases in GPU-rendered images.

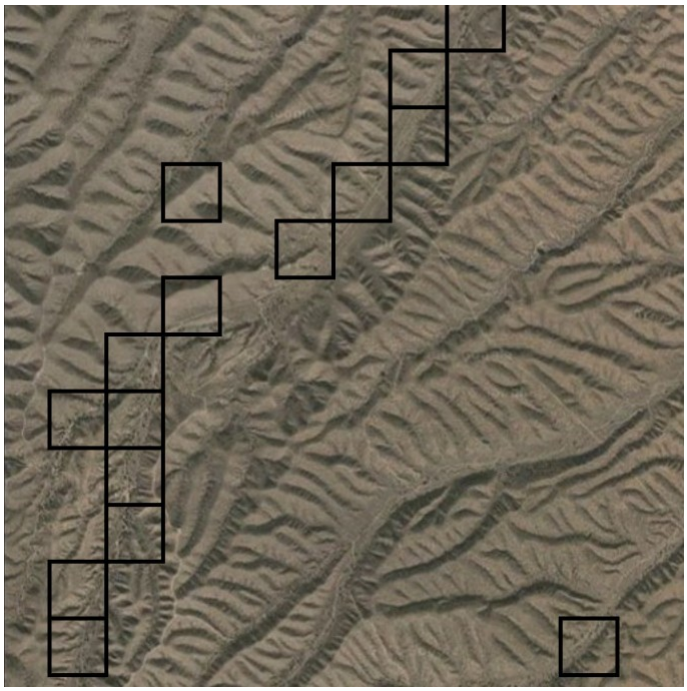
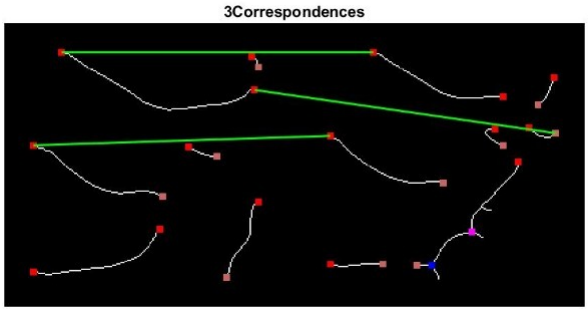


Figure 5.37: 15 Terrain blocks missed by our terrain recognition approach

One example of terrain miss is shown in Fig. 5.38. Fig. 5.38a and Fig. 5.38c are image data from camera and GPU render, respectively. By observing the aerial image, we know that there are no apparent drainage patterns in right part of the terrain block. For such an area, GRRR data fail to capture small details and as a result a flat area is generated by our rendering process. We apply MLSEC-based crease extraction on two images and present terrain valleys in Fig. 5.38b and Fig. 5.38d, respectively. Minutiae in these two crease blocks are shown in Fig. 5.38e. There are a total of 12 and 16 minutiae in these two minutiae sets, respectively. By comparing two minutiae sets, we make the observation that minutiae distribution in right parts are different due to rendering noise. We input these two minutiae sets to our matching algorithm, and obtain a total of 3 pairings. As there are a total of 28 minutiae, similarity score between these two terrain blocks is calculated as 0.21.

In this experiment, we obtain a total of 138 false positives. These false positives can be explained by the fact that there are not sufficient neighboring minutiae and crease information for a given minutiae to identify itself as size of the terrain block decreases. One example of false positive is given in Fig. 5.39. Fig. 5.39a and Fig. 5.39c are image data of different terrain

(a) Aerial image (b) Aerial crease (c) GPU image (d) GPU crease



(e) Output of matching algorithm

Figure 5.38: An example of terrain miss

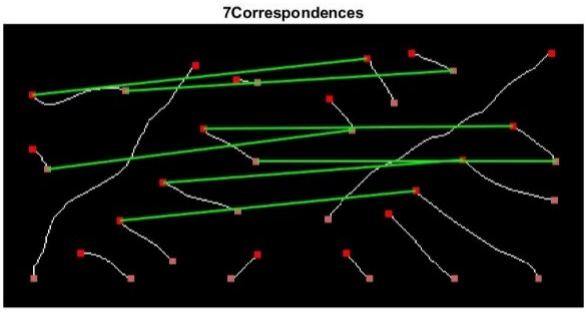
blocks. We apply MLSEC-based terrain valley extraction on these two blocks, and present the extracted terrain valleys in Fig. 5.39b and Fig. 5.39d, respectively. Minutiae in these two valley images are shown in Fig. 5.39e. There are 18 minutiae in each valley image. By comparing these two minutiae sets, we make the observation that spatial distribution of minutiae in these two minutiae sets are similar to each other. We input two minutiae sets to our minutiae-based matching algorithm, and obtain a total of 7 correspondences. Terrain similarity score between two terrain blocks is therefore calculated as 0.39.

5.4 Conclusions

We conduct two sets of experiments to study the influence of granularity of drainage patterns on performance of our terrain recognition approach. Experiment results are summarized in Table 5.15. We make the following observations from experiment results that:

- At 10m resolution, average minutiae numbers in a fixed-size terrain image of size 180 by 180 pixels over Hualapai Peak and Kings Peak are equal to 80 and 93, respectively. In

(a) Block 6 (b) Crease in block 6 (c) Block 137 (d) Crease in block 137



(e) Output of matching algorithm

Figure 5.39: An example of terrain false positive

this scenario, our terrain recognition approach successfully detect all nine blocks while producing zero false positives for both terrain areas.

- At 5m resolution, average minutiae numbers in a fixed-size terrain image of size 180 by 180 pixels over Hualapai Peak and Kings Peak are equal to 34 and 46, respectively. In this scenario, detection rate and false positive rate over Hualapai Peak are equal to 1 and 0.0016, respectively. For Kings peak, detection rate and false positive rate are equal to 0.94 and 0, respectively. Two misses over Kings Peak can be explained by weak drainage patterns features in these two terrain blocks. In summary, performance of our minutiae-based terrain recognition approach is attractive.

- At 2.5m resolution, average minutiae numbers in a fixed-size terrain image of size 180 by 180 pixels over Hualapai Peak and Kings Peak are equal to 11 and 17, respectively. In this scenario, detection rate and false positive rate over Hualapai Peak are equal to 0.83 and 0.04, respectively. For Kings peak, detection rate and false positive rate are equal to 0.89 and 0.0135, respectively.

Table 5.15: Summary of experiment results over Hualapai and Kings Peak

Metric \ Resolution	**10m**	**5m**	**2.5m**
Averge minutiae number (Hualapai)	80	34	11
Detection rate (Hualapai)	9/9 = 100%	36/36 = 100%	120/144 = 83.33%
False positive rate (Hualapai)	0/36 = 0%	1/630 = 0.16%	412/10296 = 4.00%
Averge minutiae number (Kings)	93	46	17
Detection rate (Kings)	9/9 = 100%	34/36 = 94.4%	129/144 = 89.59%
False positive rate (Kings)	0/36 = 0%	0/630 = 0.00%	138/10296 = 1.35%

Based on above observations, we conclude that

- Detection Rate: detection rates of two experiments in scenario of 2.5m resolution are equal to 83% and 89%, respectively. Compared to that of both 10m an 5m resolution, we notice obvious descrese in detection rate. This can be explained as follows. Number of drainage patterns descreas with size of terrain blocks. Under scenario of 2.5m resolution, size of terrain blocks descreses to an extent such that there exist some terrain blocks in which drainage patterns are all weak for both experiments. When deciding landmarks for navigation, we prefer terrain blocks that contain strong drainage patterns. As a result, these misses can be removed by landmark selection technique. That means, for Hualapai Peak and Kings peaks, 83 and 89 percentage of the terrains are qualified to be selected as landmarks, respectively.

- False Positive Rate: false positive rates of two experiments in scenario of 2.5m resolution are equal to 4% and 1.35%, respectively, which are higher compared to that at both 10m and 5m resolution. The increase in false positive rate can be explained by the following two facts: (1) there are not sufficient minutiae and crease information in neighborhood for a given minutiae to identify itself, and (2) correlation between contiguous terrain blocks increases. Although performance of false positive rate in scenario of 2.5m is acceptable,

it is reasonable to predict that false positive rate will continue to increase with input image resolution when sizes of inputs are fixed based on the above analysis. This can be solved by adaptively adjusting size of inputs according to flight altitudes, such that there are enough minutiae to characterize input terrain blocks. Based on the fact that average minutiae number in one terrain block at both 10m and 5m resolution is larger than 20, and average minutiae number at 2.5m resolution is smaller but approaching 20, we conclude that 20 is an appropriate threshold to identify a terrain block.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This research address the issues of UAS navigation augmentation in GPS challenged terrain environments. The proposed solution is a framework that (i) generate life-like appearance of terrain using open source map data, (ii) define an efficient terrain feature and extract them from raw terrain images, and (iii) match terrain features of aerial imagery to that of landmarks. Once a successful match is made, using a known lens model a final PVA solution can be obtained from the extrinsic matrix of the camera.

6.1.1 Minutiae Feature

We define a novel feature called minutiae for terrain recognition. Minutiae are minor details in drainage patterns, including crease endings and bifurcations. To extract drainage patterns, we design a series of filters, including diffusion filtering and MLSEC operator. We operate crease extraction approach on a set of terrain images to extract terrain valley/ridges. Experiment results prove effectiveness of the approach in accurately locating terrain creases. In addition, we make the observation from experiment results that terrain areas differ from each other in spatial distribution of both minutiae and crease.

6.1.2 Minutiae-based Terrain Matching

We define two shape descriptors, taking advantage of spatial distribution of minutiae and crease, to measure neighborhood similarity between a minutiae pair. Using these two shape descriptors, we design a matching process to identity minutiae pairings from two input minutiae sets. Similarity between two terrain blocks are defined as ratio of twice the number of minutiae

pairings to total number of minutiae. A PVA match is identified by thresholding the similarity score. We run the terrain matching approach on datasets from different areas. Experiment results prove the effectiveness of drainage patterns in terrain recognition when the requirement that minutiae number in an input terrain image is larger than 20 is satisfied.

6.1.3 Landmark Generation

Considering the fact that crease feature are highly dependent on the lighting, we design an real-time terrain generation approach. We choose GRRR data from USGS as map source for its high customizability. We implement per-pixel displacement mapping in our rendering component to generate life-like appearance of landmarks, as it would appear at that time of day, season, and orientation. A virtual sun is also modeled using directional light model and positioned with respect to the aircraft, based on the time of day. Experiment results show that our approach can accurately capture ridge-valley feature in terrain areas. Besides, our approach can render terrain image of size 512×512 pixels at average 80 frames per second.

6.2 Future Work

For many UAS applications, UASs are required to operate in complex weather conditions, such as foggy and rainy. In future work, we will conduct experiments to learn how these complex weather condition affect the performance of our navigation system.

BIBLIOGRAPHY

[1] K. Celik and A. K. Somani, "Wandless realtime autocalibration of tactical monocular cameras," *Proc. IPCV*, vol. 1, pp. 186–193, 2012.

[2] U. Niethammer, M. R. James, S. Rothmund, J. Travelletti, and M. Joswig, "UAV-based remote sensing of the Super-Sauze landslide: Evaluation and results," *Eng. Geol*, vol. 128, pp. 2–11, 2012.

[3] K. Whitehead, B. J. Moorman, and C. H. Hugenholtz, "Low-cost, on-demand aerial photogrammetry for glaciological meaurement," *The Cryosphere Discussions*, vol. 7, no. 3, pp. 3043–3057, 2013.

[4] K. Anderson and K. J. Gaston, "Lightweight unmanned aerial vehicels will revolutionize spatial ecology," *Front Ecol Envrion*, vol. 11, no. 3, pp. 138–146, 2013.

[5] J. E. Nyquist, "Unmanned aerial vehicles that even geoscience departments can afford," *Geotimes*, vol. 42, pp. 20–23, 1997.

[6] M. C. Quilter and V. J. Anderson, "A proposed method for determining shrub utilization using LA/LS imagery," *Journal of Range Management*, vol. 54, no. 4, pp. 378–381, 2001.

[7] J. Hannavy, "Encyclopedia of nineteenth-century photography," Routledge, Tylor & Francis Group, vol. 1, pp. 14–15, 2007.

[8] S. R. Herwitz, L. F. Johnson, S. E. Dunagan, et al., "Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support," *Computers and Electronics in Agriculture*, vol. 44, no. 1, pp. 49–61, 2004.

[9] J. Everaerts, "The use of unmanned aerial vehicles (UAVS) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sening and Spatial Information Sciences*, Vol. XXXVII, Part B1, Beijing, 2008.

[10] N. Haala, M. Cramer, F. Weimer, and M. Trittler, "Performance test on UAV-based photogrammetric data collection," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, pp. 7–12, 2011.

[11] M.Sauerbier, E. Siegrist, H. Eisenbeiss, and N. Demir, "The practical application of UAV-based photogrammetry under economic aspect," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, ISPRS Zurich 2011 Workshop*, vol. 38, pp. 45–50, 2011.

[12] A. D. King, "Inertial navigation–past, present, and future," *In Proceedings of the IEE Colloquium on Airborne Navigation Systems Workshop*, pp. 3/1–3/9, 1997.

[13] R. Johannessen, "The role of GPS in flight calibration," *In Proceedings of the IEE Colloquium on Current and Future Trends in Flight Calibration of Radio Navigational Aids*, pp. 9/1–9/6, 1991.

[14] P. Djederich, "Global navigation satellite systems," *In Proceedings of the IEE Airborne Navigation Systems Workshop*, pp. 1/1–1/2, 1997.

[15] T. Bernhardsen, "Geographic Information Systems: An Introduction," Wiley, Chichester, 1999.

[16] T. W. Vaneck, "Fuzzy guidance controller for an autonomous boat," *IEEE Control Systems Magazine*, vol. 17, no, 2, pp. 43–51, 1997.

[17] K. Celik, A. K. Somani, B. Schnaufer, P. Y. Hwang, G. A. McGraw, and J. Nadke, "Meta-image navigation augmenters for unmanned aircraft systems (MINA for UAS)," *Proc. SPIE 8713, Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications X*, pp. 1–15, 2013.

[18] J. P. Golden, "Terrain contour matching (TERCOM): a cruise missile guidance aid," *Proc: SPIE 0238, Image Processing for Missile Guidance*, Vol. 0238, pp. 10–18, 1980.

[19] J. R. Fountain, "Digital terrain systems," *Proc. IEE Colloquium on Airborne Navigation Systems Workshop* , 1997.

[20] J. J. Rodriguez and J. K. Aggarwal, "Matching aerial images to 3-D terrain maps," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1138–1149, 1990.

[21] D. G. Sim, R. H. Park, R. C. Kim, and S. U. Lee, "Integrated position estimation using aerial image sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence*. vol. 24, no. 1, pp. 1–18, 2002.

[22] E. Michaelsen and K. Jaeger, "A google-earth based test bed for structural image-based UAV navigation," *Proc. Int'l Conf. Information Fusion*, pp. 340–346, 2009.

[23] E. Michaelsen, K. Jaeger, D. Roschkowski, L. Doktorski, and M. Arens, Object-oriented landmark recognition for UAV-navigation. *Pattern Recognition and Image Analysis*, vol. 21, no. 2, pp. 152–155, 2011.

[24] S. J. Merhav and Y. Bresler, "On-line motion estimation from visual terrain information, part I: recursive image registration," *IEEE Trans. Aerospace and Electronic Systems*, vol. 22, no. 5, pp. 583–587, 1986.

[25] Y. Bresler and S. J. Merhav, "On-line motion estimation from visual terrain information, part II: ground velocity and position estimation," *IEEE Trans. Aerospace and Electronic Systems*, vol. 22, no. 5, pp. 588–604, 1986.

[26] R. Lerner, E. Rivlin, and P. H. Rotstein, "Error analysis for a navigation algorithm based on optical-flow and a digital terrain map," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 604–610, 2004.

[27] J. Kim and S. Sukkarieh, "Autonomous airborne navigation in unknown terrain environments," *IEEE Trans. Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1031–1045, 2004.

[28] Z. Chen and J. Samarabandu, "Using multiple view geometry within extended Kalman filter framework for simultaneous localization and map-building," *Proc. IEEE Int'l Conf. Mechatronics and Automation*, pp. 695–700, 2005.

[29] K. Celik, S. J. Chung, M. Clausman, and A. K. Somani, "Monocular vision SLAM for indoor aerial vehicles," *Proc. IEEE Int'l Conf. Intelligent Robots and Systems* , pp. 1566–1573, 2009.

[30] Z. He, R. V. Iyer, and P. R. Chandler, "Vision-based UAV flight control and obstacle avoidance," *Proc. IEEE American Control Conference*, Minneapolis, MN, 14-16 June, 2006.

[31] S. Saripalli, "State estimation for UAVs in GPS-denied environments," *Intl. Conf. Aerospae Engineering and Exhibition*, 2009.

[32] J. Zhang, Y. Wu, W. Liu, and X. Chen, "Novel approach to position and orientation estimation in vision-based UAV navigation," *IEEE Trans. Aerospace and Electronic Systems*, vol. 46, no. 2, pp. 687–700, 2010.

[33] N. Ritter and M. Ruth, "GeoTIFF format specification (GeoTIFF revision 1.0)," http://remotesensing.org/geotiff/spec/geotiffhome.html, 1995.

[34] T. G. Farr, P. A. Rosen, E. Caro, et al, "The shuttle radar topography mission," *Reviews of Geophysics*, vol. 45, no. 2, pp. 1–33, 2007.

[35] B. Rabus, M. Eineder, A. Roth, and R. Banler, "The shuttle radar topography mission-a new class of digital elevation models acquired by spaceborne radar," *ISPRS Journal of Photogrammetry & Remote Sensing*, vol. 57, no. 4, pp. 241–262, 2002.

[36] A. Koch and P. Lohmann, "Quality assessment and validation of digital surface models derived from the Shuttle Radar Topography Mission (SRTM)," *IAPRS*, Vol.XXXIII, pp. 61–68, 2000.

[37] D. Shepard, "A two-dimensional interpolation function for irregularlu-spaced data," *Proceedings of the 1968 ACM National Conference*, pp. 517–524, 1969.

[38] G. Y. Lu and D. W. Wong, "An adaptive inverse-distance weighting spatial interpolation technique," *Compuuters & Geosciences*, vol. 34, pp. 1044–1055, 2008.

[39] N. Cressie, "Statistics for spatial data," Wiley, New York, 1993.

[40] M. A. Oliver and R. Webster, "Kriging: a method of interpolation for geographical information system," *International Journal of Geographical Information Systems*, vol. 4, no. 3, pp. 313–332, 1990.

[41] S. L. Barnes and A. C. Gatreall, "Interactive spatial data analysis," Longman, Essex, England, 1995.

[42] N. Cressie, "Spatial prediction and ordinary kriging," *Mathematical Geology*, vol. 20, no. 4, pp. 405–421, 1988.

[43] K. Krivoruchko, "Spatial statistical data analysis for GIS users," ESRI Press, Redlands, California, 2011.

[44] C. DeBoor, "A practical guide to splines," Springer-Verlag, New York, 1978.

[45] C. Childs, "Interpolating surfaces in ArcGIS spatial analyst," *ArcUser*, pp. 32–35, 2004.

[46] J. C. Guarneri and R. C. Weih Jr, "Comparing methods for interpolation to improve raster digital elevation models," *Journal of the Arkansas Academy of Science*, vol. 66, pp. 77–81, 2012.

[47] R. L. Cook, "Shade trees," *Computer Graphics (SIGGRAPH'84 Proceedings)*, vol. 18, no. 3, pp. 223–231, 1984.

[48] J. Patterson, S. Hoggar, and J. Logie, "Inverse displacement mapping," *Computer Graphics Forum*, vol. 10, no. 2, pp.129–139, 1991.

[49] B. Smits, P. Shirley, and M. Stark, "Direct ray tracing of displacement mapped triangles," *11th Eurographics Workshop on Rendering*, pp. 307–318, 2000.

[50] M. Bunnell, "Adaptive tessellation of subdivision surface with displacement mapping," *In GPU Gems 2*, Parr M.,(Ed.). Addison-Wesley, Chapter 7, pp. 109–122, 2005.

[51] L. Szirmay-Kalos and T. Umenhoffer, "Displacement mapping on the GPU-state of the art," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1567–1592, 2006.

[52] W. Donelly, "Per-pixel displacement mapping with distance functions," *In GPU Gems 2*, Parr M.,(Ed.). Addison-Wesley, Chapter 8, pp. 123–136, 2005.

[53] P. E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.

[54] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," Prentice Hall, New Jersey, 2002.

[55] A. F. Sole, A. Lopez, and G. Sapiro, "Crease Enhancement Diffusion," *Computer Vision and Image Understanding*, vol. 84, pp. 41–248, 2001.

[56] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, "Ridges for image analysis," *Journal of Matchematical Imaging and Vision*, vol. 4, no. 4, pp. 353–373, 1994.

[57] A. Lopez, F. Lumbreras, J. Serrat, and J. J. Villanueva, "Evaluation of methods for ridge and velley detection," *IEEE Transation on Pattern Analysis and Matchine Intelligence*, vol. 21, no. 4, pp. 327–335, 1999.

[58] L. G. Shapiro and R. M. Haralick, "Computer and Robot Vision Volumn 1," Addison-Wesley, MA, 1992.

[59] J. J. Koenderink and A. J. Van Doorn, "Surface shape and curvature scales," *Image and Vision Computing*, vol. 10, no. 8, pp. 557–565, 1992.

[60] J. Weickert, "Anisotropic diffusion in image processing," B.G. Teubner Stuttgart, 2008.

[61] A. M. Lopez, D. Lloret, J. Serrat, and J. J. Villanueva, "Multilocal creaseness based on the level-set extrinsic curvature," *Computer Vision and Image Understanding*, vol. 77, pp. 111–144, 2000.

[62] P. W. Kwan, J. Gao, and Y. Guo, "Fingerprint matching using enhanced shape context," *Proceedings of The 21st Image and Vision Computing New Zealand (IVCNZ 2006) Great Barrier Island*, vol. 1, no. 1, pp. 115-120, 2006.

[63] M. Tico and P. Kuosmanen, "Fingerprint matching using an orientation-based minutiae descriptor," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1009–1014, 2003.

[64] J. Abramham, P. Kwan, and J. Gao, "Fingerprint matching using a hybrid shape and orientation descriptor," *State of Art in Biometrics*, DrJucheng Yang (Ed.), InTech, Chapter 2, pp. 25–56, 2011.

[65] S. A. Sudiro, M. Paindavoine, and T. M. Kusuma, "Simple fingerprint minutiae extraction algorithm using crossing number on valley structure," *Proc. IEEE Automatic Identification Advanced Technologies Workshop*, pp. 41–44, 2007.

[66] L. Hong, Y. Wan, and A. K. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777–789, 1998.

[67] R. Thai, "Fingerprint image enhancement and minutiae extraction," Technical Report, The University of Western Australia, 2003.

[68] S. Baker and R. D. Cousins, "Clarification of the use of chi-square and likelihood functions in fits to histogram," *Nuclear Instruments and Methds in Physcis Research*, vol. 221, no. 2, pp. 437–442, 1984.

[69] R. Schalkoff, "Digital Image Processing and Computer Vision," Wiley, New York, 1992.

[70] T. Wang, K. Celik, and A. K. Somani, "Characterization of mountain drainage patterns for GPS-denied UAS navigation augmentation," *Machine Vision and Application*, vol. 27, no. 1, pp. 87–101, 2016.

[71] P. J. Hardin and M. W. Jckson, "An unmanned aerial vehicle for rangeland photography," *Rangeland Ecology & Management*, vol. 58, no. 4, pp. 439–442, 2005.